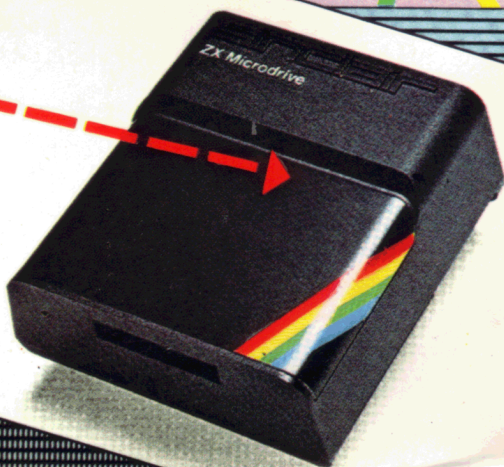
IL LIBRO DEL MICRODRIVE SPECTRUM

con dettagli tecnici dell'**INTERFACCIA ZX 1**;
del **MICRODRIVE**, della **RETE LOCALE**
e dell'**INTERFACCIA RS232.**

di Dr. Ian Logan

MICRODRIVE

ZX Microdrive

edizioni JCE

IL LIBRO DEL MICRODRIVE SPECTRUM

con dettagli tecnici dell'**INTERFACCIA ZX1**; del **MICRODRIVE**, della **RETE LOCALE** e dell'**INTERFACCIA RS232.**

di Dr. Ian Logan

traduzione di Giacomo Bortone

JACOPO CASTELFRANCHI EDITORE
Via dei Lavoratori, 124
CINISELLO BALSAMO (MI)

I N D I C E

I N D I C E   A N A L I T I C O

A

B

C

L

M

N

O

P

R

S

T

U

V

Z

P R E F A Z I O N E

Sono finalmente apparsi sul mercato i tanto attesi Microdrive, ed assieme ad essi arrivano, inseparabili, la rete di lavoro locale e l'interfaccia RS232, entrambe contenute nell'interfaccia ZX1 che serve per collegare il Microdrive allo Spectrum.

Il Microdrive e' un'unita' di memoria di massa per memorizzare programmi e file sequenziali di dati. Utilizza delle cartucce a nastro veloci che contengono circa 100 kilobyte ognuna. Il Microdrive e l'interfaccia ZX1 sono accompagnati da un manuale che spiega le caratteristiche del nuovo sistema esteso, ma si tratta di un manuale semplice, che lascia molte domande senza risposta. I capitoli di questo libro, invece, trattano dettagliatamente il sistema Spectrum esteso, il Basic esteso, il Microdrive, l'area di lavoro locale, l'interfaccia RS232, fornendo anche tutte le informazioni necessarie per programmare in linguaggio macchina.

Questo libro si pone l'obbiettivo di rispondere a tutte le domande che potrebbero venire poste dai possessori dei nuovi prodotti e non di fornire una sterile lista di programmi per dimostrare le nuove possiblita'.

Personalmente, ho scritto questo libro dopo aver partecipato in qualita' di osservatore alla stesura del Software che e' attualmente memorizzato nella ROM dell'interfaccia ZX1.

Il progetto del Microdrive ha coinvolto molte persone, ma senz'altro la parte piu' impoortante del lavoro e' stata svolta da Mr. Martin Brennan e da Mr. Ben Cheese della Sinclair Research Ltd. Devo un particolare ringraziamento a Martin, a Ben e a tutti gli altri membri del gruppo di lavoro per avermi ospitato e per aver risposto a tutte le mie domande.

Senz'altro l'introduzione del Microdrive e della rete di lavoro locale costituisce un passo avanti importante nel campo dei personal computer, ha infatti reso accessibili a molte persone delle prestazioni fino a ieri disponibili solo ad alto prezzo.

Un ringraziamento e' dovuto anche a:

- Alfred Milgrom della Melbourne House per i suoi assidui consigli.
- George Tokarski, fotografo, per le fotografie.
- Sinclair Research Ltd. per aver reso disponibili i prototipi dell' interfaccia ZX1 e del Microdrive.
- Le scuole di St. Lawrences, Skellingthorpe, Lincoln, per aver messo a disposizione un computer BBC.
- Mia moglie Liz e le mie figlie Jackie e Carolyn.

Ian Logan   Lincoln, 1983

I N T R O D U Z I O N E

Dover operare con l'interfaccia ZX1 e il Microdrive pone il possessore del nuovo sistema di fronte ai concetti di "unita' di memoria di massa" e di "collegamento tra due computer", entrambi difficili da capire in prima istanza. Fortunatamente, come in altri campi della tecnica, i principi di base sono piuttosto semplici.

In questo libro vengono esaminate una per una le possibilita' offerte dall'interfaccia ZX1 e dal Microdrive. Nella parte finale il lettore viene invitato a scrivere i propri programmi in linguaggio macchina, infatti solo cosi' il sistema puo' essere sfruttato al massimo delle sue potenzialita'.

C A P I T O L O   1

I L   S I S T E M A   S P E C T R U M   E S T E S O

L'interfaccia ZX1 puo' essere paragonata ad un "centro di comunicazione" che consente allo Spectrum di collegarsi con un Microdrive, con un altro Spectrum (attraverso una rete di lavoro locale) e con una periferica RS232.

Conseguentemente uno Spectrum dotato di un'interfaccia ZX1 puo' gestire:

dati provenienti da:
- la tastiera
- un registratore a cassette
- un Microdrive
- un altro Spectrum (attraverso la rete)
- una periferica RS232

e dati diretti verso:
- una TV
- un registratore a cassette
- un Microdrive
- un altro Spectrum (attraverso la rete)
- un'interfaccia RS232
- la ZX printer
- l'altoparlante incorporato (nello Spectrum)

Lo Spectrum usa un solo microprocessore, quindi, a parte il caso del video, dove il segnale viene generato direttamente dall'U.L.A. (Uncommitted Logic Array), il servizio di qualunque periferica di input/output assorbe completamente il calcolatore; conseguentemente, lo Spectrum non puo' svolgere nessun'altra operazione mentre si sta occupando di una periferica. Lo schema 1 illustra il sistema Spectrum esteso.

L'interfaccia ZX1 viene ora analizzata esaminando separatamente in dettaglio:

1) i connettori;

2) la gestione della memoria;

3) l'elettronica.

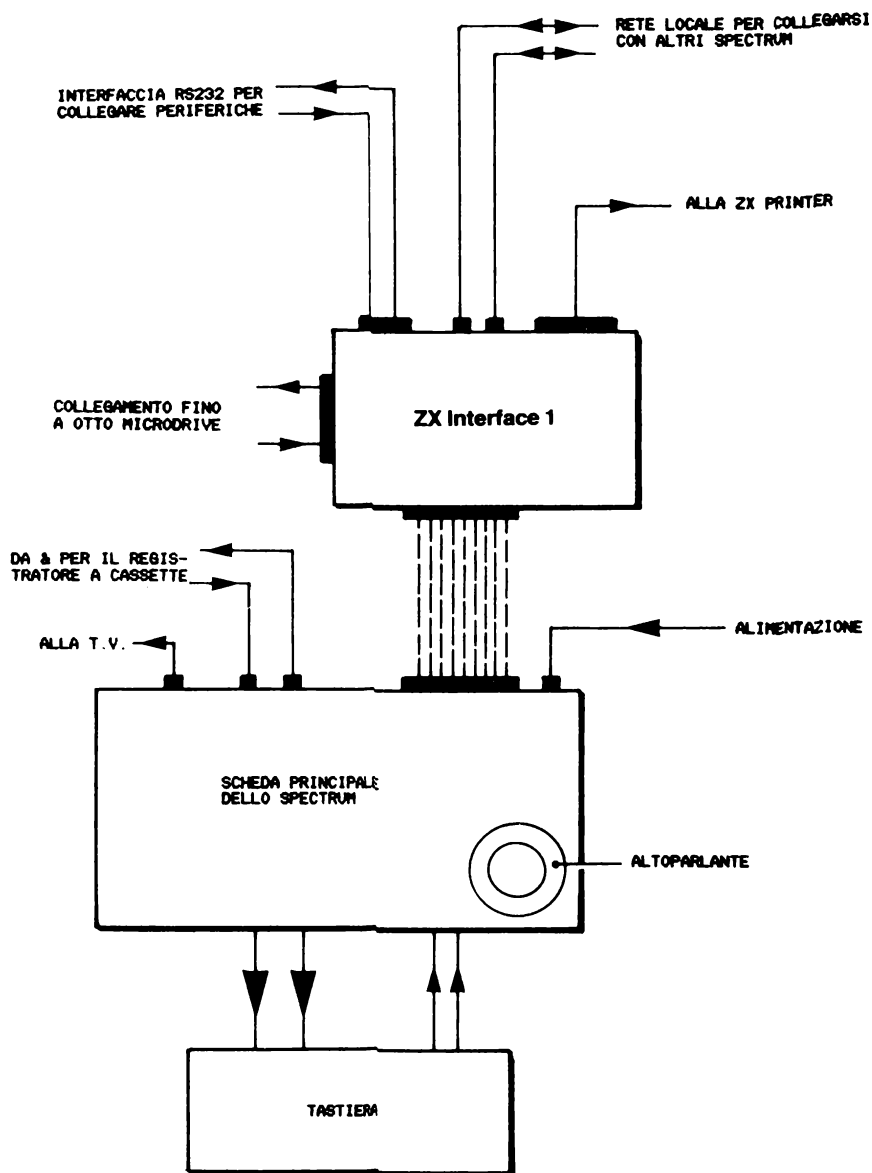



FIG.1. SCHEMA A BLOCCHI DEL SISTEMA SPECTRUM ESTESO

I CONNETTORI

Sull'interfaccia ZX1 sono presenti cinque connettori. Essi sono:

a. IL CONNETTORE PIATTO PRINCIPALE

E' un connettore a 2x28 poli che collega l'interfaccia ZX1 allo Spectrum. Le linee piu' importanti che attraversano il connettore sono:

- il bus degli indirizzi a 16 linee;
- il bus dei dati a 8 linee;
- la linea di selezione della ROM;
- l'alimentazione (+9v., +5v. e 0v.);
- il segnale M1 (caricamento di istruzione),

ma anche le altre linee di controllo sono importanti per il corretto funzionamento dell'interfaccia.

b. IL CONNETTORE DI ESTENSIONE

Contiene le stesse 56 linee del connettore dello Spectrum. Serve per collegare la stampante ZX Printer oppure altre periferiche.

c. IL CONNETTORE PER IL MICRODRIVE

E' un connettore a 14 poli. Contiene i segnali:

- 0v.
- +9v.
- due linee di dati bidirezionali
- R/W (lettura/scrittura)
- cancellazione
- due linee di controllo
- protezione contro la scrittura
- cinque linee non utilizzate, collegate a massa.

I dati vengono trasmessi tra l'interfaccia e il Microdrive in forma seriale, con i bit trasmessi alternativamente sulle due linee. La porta per gestire i dati e' la porta E7h.
Le due line di controllo sono usate per selezionare il Microdrive richiesto e per avviare o fermare il suo motore. Queste linee sono gestite dalla porta EFh, che gestisce anche le linee di cancellazione e di protezione contro la scrittura.

d. IL CONNETTORE RS232

E' un connettore a vaschetta a nove poli. Contiene le linee;

per la trasmissione dei dati:	-Pin 3: Data out (RXdata); -Pin 4: Data terminal ready (DTR);
per la ricezione dei dati:	-Pin 2: Data in (TXdata); -Pin 5: Clear to send (CTS);
e la massa:	-Pin 7: 0v.

Durante la trasmissione lo Spectrum invia un byte di dati sulla linea RX solo dopo che DTR viene trovato a livello alto. I dati vengono ricevuti sulla linea TX solo dopo che lo Spectrum ha segnalato che e' pronto a ricevere portando a livello alto la linea CTS.
L'interfaccia RS232 usa il bit 0 della porta F7h per la trasmissione seriale dei dati. Lo stato del segnale DTR puo' essere rivelato leggendo il bit 3 della porta EFh e il segnale CTS puo' essere portato a livello alto settando il bit 4 della stessa porta.

e. IL CONNETTORE DELLA RETE DI LAVORO LOCALE

Ci sono due connettori per il collegamento con altri Spectrum. Essi sono esattamente equivalenti e sono in coppia soltanto per permettere di collegare piu' Spectrum in cascata, cioe' per permettere di collegare due calcolatori ad un solo Spectrum senza bisogno di sdoppiare il cavo.

La rete usa solo due linee, la linea del segnale e la massa. La linea del segnale si setta utilizzando il bit 0 della porta F7h.

Si noti che la rete prevale sull'interfaccia RS232. L'interfaccia RS232 viene attivata al posto della rete quando CTS viene portato a livello alto (bit 4 della porta EFh), sia durante la trasmissione che durante la ricezione.

GESTIONE DELLA MEMORIA

Lo Spectrum e' dotato di una ROM da 16k che occupa l'area di memoria dall'indirizzo 0 (0000h) all'indirizzo 16383 (3FFFh). In questa ROM si trova il sistema operativo, l'interprete Basic, l'aritmetica in virgola mobile e il set dei caratteri. Questa ROM pero' non contiene il software necessario alla rete di lavoro locale, ai Microdrive e all'interfaccia RS232.

E' quindi necessaria un'estensione della ROM standard. Esiste infatti una seconda ROM di 8K, che chiameremo "fantasma", contenuta nell'interfaccia ZX1. Tramite un sistema di selezione delle ROM lo Spectrum puo' utilizzare in ogni momento una ROM oppure l'altra, alternativamente.

Nella ROM fantasma dell'interfaccia ZX1 sono contenute:

- un'estensione di quella parte del Basic che controlla la correttezza della sintassi e dei comandi, per permettere che vengano accettati anche i comandi privi di significato senza espansione;
- le routine che eseguono i nuovi comandi;
- le routine di input e output per il Microdrive, per la rete di lavoro locale e per l'interfaccia RS232.

Per rendersi conto dell'effetto della ROM fantasma, basta osservare che alcune espressioni che prima avrebbero causato messaggi di errore vengono eseguite in modo corretto.

Per esempio il comando CAT1 che pur essendo previsto avrebbe dato un messaggio di errore, produce un risultato ben preciso con l'interfaccia ZX1 e un Microdrive con inserita una cartuccia. La ROM fantasma dell'interfaccia ZX1 viene abilitata eseguendo un caricamento di istruzione (FETCH) ad una delle due locazioni 0008h o 1708h, la ROM standard viene ripristinata eseguendo un caricamento di istruzione alla locazione 0700h.

A questo punto e' opportuno considerare in dettaglio due routine della ROM fantasma.

---

a.LA ROUTINE INSER-

La prima volta che la ROM fantasma viene inserita nel sistema, subito dopo l'accensione o dopo l'esecuzione di un comando NEW, vengono create 58 nuove variabili di sistema.

La creazione delle nuove variabili e la loro inizializzazione viene gestita dalla routine INSER-. Vengono inizializzate:

NTSTAT a 1 - numero di stazione = 1;
IOBORD a 0 - colore di input/output = NERO;
BAUD a 12 - baud rate = 9600.

---

b. LA RUOTINE CALBAS

Questa routine consente alla ROM fantasma di utilizzare delle routine contenute nella ROM principale dello Spectrum. La sua caratteristica piu' importante e' che tutti i registri del microprocessore vengono salvati prima della chiamata alla ROM principale e ripristinati teminata l'esecuzione della routine chiamata.

La routine CALBAS chiama a sua volta la routine di gestione della ROM (locazioni 23737-23746) contenuta nella zona delle variabili di sistema; questa effettua effettivamente la chiamata della subroutine richiesta nella ROM principale.

Ecco il listato della routine di gestione della ROM:

```
        ORG   23737
H-L     EQU   23738

SBRT    LD    HL,...      ;valore necessario per HL
        CALL  ...         ;indirizzo della subroutine
                          ;da chiamare
        LD    (H-L),HL    ;salva il nuovo valore di HL
        RET               ;ritorno
```

Le locazioni H-L sono usate per memorizzare il valore della coppia di registri HL che viene alterato durante le operazioni di scambio delle ROM. Il valore memorizzato e' quindi ripristinato nei registri HL al momento opportuno.

CODICI DI CHIAMATA

Le routine della ROM fantasma possono essere usate dal linguaggio macchina mediante una serie di codici di chiamata. Questi codici devono essere specificati dopo l'istruzione RST 0008h e vengono descritti nel capitolo relativo al linguaggio macchina.

ELETTRONICA

E' opportuno distinguere cinque funzioni differenti svolte dal circuito elettronico dell'interfaccia, anche senza entrare nei dettagli del funzionamento elettrico.

1.MECCANISMO DI GESTIONE DELLA ROM

La ROM fantasma deve essere inserita e disinserita a seconda delle necessita'. Questa operazione coinvolge la decodifica di opportuni indirizzi e del segnale M1. Il risultato e' che la linea di selezione della ROM (ROMCS) e' portata a livello alto quando la ROM fantasma e' inserita e a livello basso quando viene disinserita.

2. GESTIONE DEGLI INDIRIZZI DELLE PORTE

Le periferiche collegate all'interfaccia ZX1 sono selezionate utilizzando le linee di indirizzi A3 e A4 con le istruzioni IN e OUT. Una parte dell'elettronica si occupa di decodificare opportunamente i segnali utili.

3. GESTIONE DI DATI DA E VERSO IL MICRODRIVE

I dati che vengono trasmessi dallo Spectrum al Microdrive devono passare attraverso un convertitore da parallelo a seriale. In piu' i bit devono essere divisi tra due linee di dati. La trasformazione opposta deve avvenire durante il trasferimento in senso contrario.

4. GESTIONE DI DATI DA E VERSO LA RETE DI LAVORO LOCALE

La trasmissione di dati e' molto semplice, ma la ricezione degli stessi richiede che l'hardware sia in grado di distinguere i segnali fasulli eventualmente presenti sulla linea seriale. Di questo si tratta piu' in dettaglio nel capitolo che riguarda la rete.

5. GESTIONE DI DATI DA E VERSO L'INTERFACCIA RS232

In questo caso l'elettronica si occupa solo di assicurare i corretti livelli di segnale.

La maggior parte dei circuiti che svolgono le operazioni sopra descritte sono contenuti in un singolo ULA (Uncommitted Logic Array).

C A P I T O L O 2

I L B A S I C E S T E S O

L'interfaccia ZX1 incorpora un'espansione del Basic residente nello Spectrum. Una volta collegata l'interfaccia, saranno riconosciuti ed eseguiti correttamente comandi che prima non potevano essere utilizzati.

L'estensione coinvolge i seguenti comandi, gia' presenti sulla tastiera dello Spectrum:

- FORMAT
- OPEN#
- CAT
- ERASE
- MOVE
- SAVE, LOAD, VERIFY & MERGE
- CLS & CLEAR

Di seguito viene spiegata la funzione e la sintassi di ogni comando.

F O R M A T

La sintassi del comando FORMAT e':

- FORMAT
- indicatore di periferica (M,m,N,n, T,t,B o b)
- un separatore (, oppure ;)
- un'espressione numerica

e in piu', se richiesto:

- un separatore (, oppure ;)
- un'etichetta (ovvero un nome)

Il comando FORMAT ha una funzione diversa per ognuna delle connessioni dell'interfaccia ZX1, cioe' per il

Microdrive, la rete e l'interfaccia RS232.

FORMAT CON IL MICRODRIVE

Un comando FORMAT con "M" o "m" come indicatore di periferica e' diretto verso il Microdrive e causa la formattazione di una cartuccia. La cartuccia inserita nel Microdrive specificato viene pulita da tutti i dati precedentemente memorizzati e viene etichettata con il nome fornito.

Ecco alcuni esempi di comando FORMAT di questo tipo:

FORMAT "M";1;"primo"
- cancella la cartuccia nel Microdrive 1 e le assegna il nome "primo"

FORMAT A$;B,C$
- A$ puo' contenere "M" o "m"; B e' un numero tra 1 e 8 e C$ e' una stringa contenente tra 1 e 10 caratteri.

Nel caso che uno dei parametri forniti sia fuori dall'intervallo consentito o nel caso che non ci sia una cartuccia adatta alla formattazione (non protetta) inserita nel Microdrive specificato, vengono visualizzati i relativi messaggi di errore.

FORMAT CON LA RETE

Un comando FORMAT avente come indicatore di periferica "N" o "n" assegna allo Spectrum su cui viene eseguito il numero di stazione indicato. All'accensione lo Spectrum e' inizializzato automaticamente come stazione 1.

Ecco alcuni esempi di un comando FORMAT di questo tipo:

FORMAT "n";2
- il numero di stazione diventa 2

FORMAT A$;B
- e' equivalente se A$ contiene "n" o "N" e B e' un numero tra 1 e 64.

Se una delle espressioni e' fuori dall'intervallo consentito viene visualizzato il relativo messaggio di

errore, ma se sono presenti anche un altro separatore e un'etichetta, questi vengono ignorati senza alcun messaggio errore.

FORMAT CON L'INTERFACCIA RS232

Un comando FORMAT avente come indicatore di periferica "T", "t", "B" o "b" inizializza la baud rate per le operazioni di input/output. All'accensione la baud rate e' automaticamente settata a 9600. Con il comando FORMAT questo valore puo' essere cambiato in uno dei valori standard tra 50 e 19200, cioe':

50, 110, 300, 600, 1200, 2400, 4800, 9600, 19200.

Esempi di un comando FORMAT di questo tipo sono:

FORMAT "T";1200
- setta la baud rate a 120
FORMAT A$,B
- puo' essere usato se A$ e' "T", "t","B" o "b" e B ha un qualunque valore numerico.

Se l'espressione di indicatore di periferica non viene riconosciuta o se si cerca di dare alla baud rate un valore superiore a 65535, vengono visualizzati i relativi messaggi di errore.

Per quanto teoricamente si possono usare solo nove valori diversi per la baud rate, nel caso che vengano specificati valori non consentiti non si ottiene un messaggio di errore ma viene selezionata la baud rate standard inferiore al valore specificato (naturalmente tenendo 50 come limite minimo). Per esempio:

FORMAT "t",25000 setta la baud rate a 19200
FORMAT "t",5 setta la baud rate a 50

Si noti infine che non c'e' differenza nell'usare "T" o "B" come indicatore di periferica.

O P E N

La sintassi del comando OPEN# e':

- OPEN#
- un numero di flusso (normamente tra 4 e 15)
- un separatore (, oppure ;)
- un'indicatore di periferica (M,m,N,n,T,t,B,b)

e, se richiesto:

- un separatore (, oppure ;)
- un'espressione numerica

e, se richiesto:

- un separatore (, oppure ;)
- un'etichetta

Nello Spectrum un comando OPEN# ha la funzione di associare la periferica specificata al flusso specificato. Questa operazione e' necessaria per controllare la periferica attraverso i comandi: PRINT#n, INPUT#n, INKEY$#n; dove n e' il numero di flusso associato alla periferica.

All'accensione lo Spectrum associa la tastiera al flusso 0 e al flusso 1, lo schermo televisivo al flusso 2 e la stampante ZX Printer al flusso 3. Per quanto sia possibile cambiare queste assegnazioni, cio' non e' di alcuna utilita' pratica.

Per aprire un flusso il computer svolge due operazioni distinte: dapprima sistema nell'area delle informazioni di canale i dati di canale necessari per la periferica, e quindi pone nella zona dei dati dei flussi lo spiazzamento dell'indirizzo (la differenza tra l'indirizzo base nei dati di canale e la variabile di sistema CHANS).

Viene ora discusso l'uso del comando OPEN per associare un flusso ai Microdrive, alla rete e all'interfaccia RS232.

OPEN# CON IL MICRODRIVE

Un comando OPEN# avente come indicatore di periferica "M" oppure "m" associa il file specificato su un certo Microdrive al flusso specificato.
Esempi di un comando di questo tipo sono:

OPEN#4;"M",1;"primo file"
- con il quale il file di nome "primo file" sul Microdrive 1 viene associato al flusso 4.
OPEN#A;B$,C,D$(2)
- A puo' avere un valore tra 0 e 15; B$ e' l'espressione "M" o "m"; C ha un valore tra 1 e 8 e D$(2) e' una stringa lunga da 1 a 10 caratteri.

Dopo l'esecuzione di un comando OPEN# di questo tipo il file specificato viene automaticamente aperto, per la scrittura se non esisteva in precedenza, per la lettura se esisteva gia'. Con i Microdrive non e' possibile allungare un file preesistente sotto il controllo del Basic, ma, per lo scopo, e' necessario riscrivere un nuovo file e cancellare quello vecchio.
Se il flusso specificato e' gia' aperto viene visualizzato un messaggio di errore (ma non con i flussi 0-3), se invece non c'e' sufficiente memoria per allungare l'area delle informazioni di canale di 595 byte viene visualizzato il messaggio di errore "out of range".
Si noti che l'uso del comando OPEN# per creare un nuovo file non produce realmente la scrittura del file stesso sul Microdrive. Il file viene creato all'atto della scrittura fisica sulla cartuccia, cioe' quando vengono scritti piu' di 512 caratteri o viene chiuso il flusso associato.

OPEN# E LA RETE

Un comando OPEN# avente come indicatore di periferica "N" oppure "n", associa il flusso specificato alla trasmissione o alla ricezione di dati sulla rete da, oppure verso, uno Spectrum che usa il numero di stazione specificato.

Esempi di un comando OPEN# di questo tipo sono:

OPEN#4;"N",44
- che destina il flusso #4 alle comunicazioni con uno Spectrum con numero di stazione 44.
OPEN#A,B$,C
- A puo' avere un valore tra 0 e 15; B$ e' la stringa "N" o "n" e C ha un valore tra 0 e 64.

Analogamente agli altri casi, se il flusso specificato e' gia' aperto, oppure se c'e' insufficiente memoria per poter aggiungere 276 byte, se l'area delle informazioni di canale o una delle variabili e' fuori dall'intervallo consentito, vengono visualizzati i relativi messaggi di errore. Il comando OPEN# non determina se il flusso deve essere usato per trasmettere o per ricevere dati, ma una volta che il buffer contiene dei dati, in trasmissione oppure ricevuti da un'altra stazione, non e' piu' possibile sfruttarlo per l'operazione contraria, cioe' i dati ricevuti possono essere soltanto letti e i dati da trasmettere possono essere soltanto trasmessi.

OPEN# CON L'INTERFACCIA RS232

Un comando OPEN# contenente come indicatore di periferica "T", "t", "B" o "b" associa il flusso specificato all'interfaccia RS232, per input ed output. Il comando non cambia la baud rate. Esempi validi sono:

OPEN#4;"T"
- il flusso 4 e' assegnato all'interfaccia RS232 e i dati in input e output vengono trattati come testo;
OPEN#A,B$
- A puo' avere un valore tra 0 e 15 e, se B$ e' l'espressione "B" o "b", i dati trasmessi o ricevuti vengono trattati come dati binari.

Se uno dei parametri e' fuori dall'intervallo consentito viene visualizzato il messaggio di errore "out of range". E' difficile, per quanto possibile, che si ottenga un messaggio "out of memory" aprendo un flusso per l'interfaccia RS232 dato che questo richiede soltanto 11 byte di dati di canale.

C A T

La sintassi del comando CAT e':

- CAT

se necessario:

- un simbolo #
- un numero di flusso
- un separatore (, oppure ;)

e sempre:

- un numero di Microdrive

Un comando CAT produce semplicemente la lista dei primi 50 file non protetti trovati nella cartuccia inserita nel Microdrive specificato e stampa la quantita' di spazio ancora disponibile sulla cartuccia stessa. Si ricordi che i nomi dei file che cominciano con CHR$ 0 sono protetti e vengono ignorati dal comando CAT.

Se non viene dato alcun numero di flusso la lista viene inviata al flusso 2, cioe' normalmente allo schermo televisivo. Esempi dei piu' comuni comandi CAT sono:

CAT 1
- che produce la lista dei file non protetti della cartuccia del Microdrive 1 sulla periferica associata al flusso numero 2 (normalmente il video);

CAT#A;B
- A puo' avere un valore tra 0 e 15 e B un valore tra 1 e 8.

Se uno dei parametri e' fuori dall'intervallo consentito, oppure se il flusso che deve essere usato per l'output non e' stato aperto, o se non c'e' la cartuccia nel Microdrive specificato, viene visualizzato il relativo messaggio di errore.

Si noti che un comando formato soltanto dalla parola CAT viene accettato ma produce un errore quando il computer cerca di eseguirlo. Questo avviene perche' la sintassi del comando viene accettata dalla ROM originale della Specrum e non puo' quindi essere intercettato dalla

ROM dell'interfaccia ZX1 fino a quando non viene eseguito. Usare CAT e ENTER e' un modo facile per verificare se l'inserzione della ROM fantasma funziona correttamente.

E R A S E

La sintassi del comando ERASE e':

- ERASE
- un'espressione di device (M o m)
- un numero di Microdrive
- un' etichetta

Un comando ERASE cancella il file specificato nel Microdrive specificato. I blocchi di dati che vengono cancellati diventano liberi e possono essere utilizzati da un altro file.

Esempi di possibili comandi di ERASE sono:

ERASE "M";1;"primo file"
- il file chiamato "primo file" viene cancellato dalla cartuccia inserita nel Microdrive 1.

ERASE A$;B,C$
- dove A$ e' l'espressione stringa "M" o"m"; B ha un valore nell'intervallo 1-8 e C$ contiene il nome di un file.

Se uno dei parametri e' fuori dall'intervalo consentito, o se non e' inserita una cartuccia nel Microdrive specificato, o se il file non esiste, viene visualizzato il relativo messaggio di errore.

M O V E

La sintassi del comando MOVE e':

- MOVE
- un simbolo #
- un numero di flusso
- la parola chiave TO

- un simbolo #
- un numero di flusso

Dove il flusso che precede la parola chiave "TO" e' la sorgente dei dati e quello che la segue e' la destinazione dei dati. Entrambi possono essere sostituiti con gli specificatori di canale. Cioe':

- un indicatore di periferica

e se richiesto:

- un separatore
- un'espressione numerica

e se richiesto:

- un separatore
- un'etichetta

Un comando MOVE ha l'effetto di prelevare i dati dal flusso/canale sorgente e inviarli al flusso/canale destinazione. Il tempo impiegato da un comando MOVE che usa per indicare la sorgente e la destinzione un numero di flusso non e' piu' lungo di quello indicato dallo stesso comando che usa gli specificatori di canale.
Esempi di comandi MOVE sono:

MOVE #4 TO #5
- dove i dati vengono prelevati dal flusso #4 e mandati al flusso #5
MOVE "N";20 TO "M";1;"dati rete"
- dove i byte dei dati ricevuti dalla stazione 20 dalla rete sono mandati al file "dati rete".

Se uno dei parametri e' fuori dall'intervallo specificato, oppure se il flusso e' chiuso, se non c'e' una cartuccia nel Microdrive specificato, se non puo' essere reperita la sorgente dei dati, o se il file di destinazione esiste gia', viene visualizzato il relativo messaggio di errore.

Il comando MOVE deve essere usato con file di dati di dimensioni finite cioe' che recano alla fine l'indicazione di fine file ("end of file"). Se questa indicazione e' assente il file e' considerato di dimensioni infinite e cio' puo' provocare dei problemi, come pure puo' provocare dei problemi il trasferimento di programmi Basic e di matrici.

S A V E ,   L O A D ,   V E R I F Y ,   M E R G E

Questi quattro comandi sono usati per gestire programmi Basic o matrici.
La sintassi dei comandi e':

- SAVE/LOAD/VERIFY/MERGE
- un asterisco
- un'espressione di device (M,m,N,n,T,t,B,b)

e se richiesto;

- un separatore
- un'espressione numerica

e se richiesto:

- un separatore
- un'etichetta

e se richiesta una delle seguenti estensioni:

- (se non c'e' niente si intende un programma Basic);

oppure:

- LINE
- un'espressione numerica (il programma parte automaticamente dalla linea indicata quando viene caricato);

oppure:

- DATA (matrice di variabili gestita dal Basic)

oppure:

- CODE (blocco di dati binari)

e se richiesto:

- un'espressione numerica (l'indirizzo di partenza)

e se richiesto:

- un separatore (,)
- un'espressione numerica (la lunghezza del blocco)

oppure:

- SCREEN$ (che sostituisce CODE e i parametri nel caso del display file e del file degli attributi per memorizzare l'immagine video).

Le ultime 2 espressioni numeriche sono richieste nell'uso di SAVE ... CODE. L'asterisco dopo la parola chiave indica alla routine del sistema che controlla la sintassi che non si tratta di un comando diretto alla cassetta. I comandi SAVE, LOAD, VERIFY e MERGE hanno lo stesso significato in un sistema espanso o in un sistema non espanso, con l'unica differenza che i programmi e le matrici trasmessi sulla rete all'interfaccia RS232 sono privi di nome.

Esempi di possibili comandi SAVE sono:

SAVE* "m",1,"primo programma"
- memorizza il programma Basic in memoria nel file "primo programma" nella cartuccia inserita nel Microdrive 1.
SAVE* "n";2
- trasmette il programma Basic in memoria attraverso la rete alla stazione 2
SAVE* "b" SCREEN$
- l'immagine mostrata sullo schermo viene trasmessa attraverso l'interfaccia RS232. In questo caso l'uso dell'indicatore di periferica "t" sarebbe stato scorretto e avrebbe prodotto un errore, dato che "t" e' riservato ai testi.

Esempi di possibili comandi LOAD sono:

LOAD*"m";4,"quarto programma"
- carica il file "quarto programma" solo se esiste e si tratta di un programma in Basic
LOAD*"n";33
- carica un programma inviato dalla stazione 33
LOAD*"b"
- carica un programma Basic attraverso l'interfaccia RS232.

Si noti che non e' possibile distinguere i dati provenienti dall'interfaccia RS232 e dalla rete, quindi se i dati vengono caricati come programma Basic per errore, si otterra' un arresto (crash) del sistema invece del messaggio di errore "wrong file type" (tipo di file errato) del Microdrive.

Esempi di possibili comandi VERIFY sono:

VERIFY*"M",4;"quarto programma"
- confronta il file "quarto programma" con il programma in memoria

VERIFY*"N";33
- confronta la "seconda" copia del programma Basic trasmesso dalla stazione 33 con la prima per assicurarsi che questa sia stata caricata correttamente

VERIFY*"B" CODE 32000,256
- verifica che i 256 byte a partire dall'indirizzo 32000, siano uguali a quelli ricevuti attraverso l'interfaccia RS232.

Esempi di possibili comandi MERGE sono:

MERGE*"m";1;"parte due"
- aggiunge al programma Basic in memoria il programma Basic "parte due"

MERGE*"n";33
- aggiunge al programma in memoria il programma ricevuto dalla stazione 33

MERGE*"b"
- aggiunge al programma in memoria il programma ricevuto attraverso l'interfaccia RS232

Si ricordi che MERGE puo' essere impiegato soltanto con i programmi Basic e le loro variabili; diversamente si ha un messaggio "MERGE error". I programmi memorizzati con l'opzione LINE (partenza automatica) possono essere caricati con MERGE solo da cassetta.

C L S #

Il comando CLS# svolge le operazioni del ben noto comando CLS ma in piu' ripristina le condizioni iniziali del video; cioe' pone:

INK = nero
PAPER = bianco
BORDER = bianco
INVERSE = no
BRIGHT = no
OVER = no
FLASH = no

Inoltre il comando CLS# cancella completamente lo schermo.

C L E A R #

Il comando CLEAR# pulisce l'area di dati dei flussi. Tutti i flussi vengono chiusi, i flussi dallo 0 al 3 vengono assegnati alle loro periferiche normali, i flussi dal 4 al 15 vengono resettati. Se in qualche buffer sono presenti dati non ancora inviati, essi sono cancellati (e non inviati come avverrebbe col comando CLOSE#) e l'area dei dati del canale e' ridotta alla minima dimensione possibile.

Infine l'utente puo' definire i propri comandi Basic estesi analogamente a CLS# e CLEAR#. Nel capitolo del linguaggio macchina verra' spiegato come puo' essere fatto cio'.

UN APPUNTO SULLA GESTIONE DEI FLUSSI

Nello Spectrum il flusso di output rimane selezionato tra un comando e l'altro. Questo non e' importante in uno Spectrum normale, ma puo' provocare la mancata esecuzione di alcuni comandi in un sistema esteso. Per esempio:

```
CLS#:PAPER2:CLS
```
- pulisce lo schermo e lo fa diventare rosso

invece:

```
CLS#:SAVE*"n";0:PAPER2:CLS
```
- non lo fa

Il rimedio e' di riselezionare lo schermo prima di usare PAPER, CLS, INK, ecc. Per esempio con:

```
CLS#:SAVE*"n";0:PRINT;:PAPER2:CLS
```
- che riseleziona lo schermo prima di PAPER2, anche se il comando "PRINT;" non ha alcun effetto apparente.

C A P I T O L O   3

I L   M I C R O D R I V E

NOTA DELL'AUTORE

La Sinclair Research Ltd. ha chiesto che non fossero pubblicate alcune informazioni riservate riguardanti il Microdrive, quindi in questo capitolo alcuni argomenti non sono approfonditi come avrebbero potuto. Quanto pubblicato e' comunque completo e di grande interesse.

INTRODUZIONE

Il sistema del Microdrive e' essenzialmente un sistema a mini-cassette e puo' essere considerato diviso in 4 parti che vengono esaminate individualmente.

IL MICRODRIVE

Il Microdrive puo' essere considerato come un registratore a mini-cassette speciale, dato che usa delle cassette progettate appositamente. Otto di queste unita' possono essere collegate assieme e collegate allo stesso Spectrum con una sola interfaccia ZX1. Ogni Microdrive ha un motore, una testina a due tracce per lettura, scrittura, cancellazione e l'elettronica necessaria per far funzionare il tutto; sul lato frontale si trova una fessura per ospitare una cartuccia.

LA CARTUCCIA DEL MICRODRIVE

Una cartuccia contiene un unico pezzo di nastro lungo circa 5 metri e largo 1,5 mm. che e' giuntato in un punto e sembra quindi continuo. Il nastro e' arrotolato in un'unica bobina, dal cui centro viene estratto per

passare davanti alle testine e venire poi trascinato da una puleggia, andando a riavvolgersi sull'esterno della bobina stessa.

IL CONNETTORE DEL MICRODRIVE

La prima unita' viene collegata all'interfaccia ZX1 tramite un cavo piatto a 16 conduttori. Le unita' successive vengono collegate l'una all'altra mediante connettori con lo stesso numero di poli.

L'INTERFACCIA ZX1

E' un scatola che si attacca sotto lo Spectrum e contiene:

a.) Il software necessario per aggiungere al Basic le funzioni per controllare il sistema dei Microdrive, la rete e l'interfaccia RS232.

b.) L'hardware necessario a gestire i segnali di controllo e a convertire i dati dalla forma seriale a quella parallela, e viceversa, durante la trasmissione tra lo Spectrum e le unita' esterne.

USO DEL MICRODRIVE

In media una cartuccia di Microdrive puo' contenere 90 kilobyte, che possono essere costituiti da programmi o da file di dati. Una singola cartuccia puo' contenere fino a circa 180 file contemporaneamente, che possono essere sia programmi che dati. Ogni file deve avere un nome diverso da tutti gli altri e tale nome deve essere lungo da 1 a 10 caratteri. Sono permessi tutti i caratteri comprese le parole chiave. Il nastro della cartuccia e' diviso in settori di 512 byte ognuno. Le operazioni di lettura e di scrittura non possono mai riguardare meno di un settore e un file occupa un numero di settori sufficiente per contenere tutti i byte che lo compongono.

NOTA

Una cartuccia contiene un nastro di circa 5 m. (200 pollici) e dato che ci sono circa 200 settori sul nastro, si trova circa un settore ogni 2,5 cm. (un pollice) quindi la densita' dei dati e' di circa 500 byte/pollice (vedi oltre per maggiori dettagli).

Un file occupa un solo settore se e' piu' corto di 512 byte, e piu' settori se e' piu' lungo. Si noti che un settore puo' contenere un solo file, anche se non viene utilizzato completamente. I settori che contengono ognuna delle parti componenti un file costituiscono i record di questo file e sono numerati progressivamente (0,1,2....n). Le possibilita' offerte nella gestione del Microdrive sono esaminate in questo capitolo.

FORMATTAZIONE DI UNA CARTUCCIA

Una cartuccia nuova deve essere preparata per l'uso eseguendo il comando:

FORMAT"m";1;"...nome..."
-con la cartuccia nel drive 1

Questo comando svolge tre funzioni:

1.) cancella completamente il nastro;

2.) lo formatta, cioe' lo divide in settori e verifica la formattazione;

3.) scrive il nome della cartuccia nell'intestazione di ogni settore.

CATALOGO DI UNA CARTUCCIA

La lista dei contenuti di una cartuccia puo' essere ottenuta in qualsiasi momento usando:

CAT1
-con la cartuccia inserita nel Microdrive 1. Il catalogo appare sullo schermo; invece con:

CAT #4;1
-il catalogo viene stampato sul flusso #4.

Il catalogo e' formato da:

- il nome della cartuccia
- una lista dei primi 50 nomi di file trovati nella cartuccia (sistemati in ordine alfabetico)
- il numero di kilobyte liberi nella cartuccia.

Si noti che i nomi dei file vengono visualizzati senza alcuna indicazione se si tratti di programmi o di dati e i nomi che cominciano con CHR$ 0 non vengono rivelati.

CANCELLAZIONE DI UN FILE

Qualunque file puo' essere cancellato usando il comando:

ERASE"m";1,"...nome..."
- con la cartuccia inserita nel Microdrive 1

Questa operazione e' piuttosto lenta (circa 40 sec.) se non e' presente l'indicatore di fine file (file non chiusi, ecc.).

GESTIONE DI PROGRAMMI BASIC, MATRICI E BLOCCHI DI DATI

MEMORIZZAZIONE DI UN PROGRAMMA

I file contenenti programmi (chiamati in questa trattazione file-programmi) sono creati con l'uso del comando SAVE, per esempio:

SAVE*"m",1,"prog uno"
- crea un file-programma di nome "prog uno" sulla cartuccia nel Microdrive 1
SAVE*"m";2;"schermo"SCREEN$
- crea un file-programma di nome "schermo"

nella cartuccia del Microdrive 2
SAVE*"m";3;"matrice1"DATA A ()
- crea un file-programma di nome "matrice1" nella cartuccia del drive 3

Da questi esempi emerge che i file creati con SAVE (file-programmi) possono contenere: programmi Basic e le loro variabili, matrici o blocchi di dati binari (come le cassette).

Se il programma Basic deve essere lanciato automaticamente al caricamento deve essere memorizzato con l'opzione LINE. Per esempio:

SAVE*"m";1;"programma due" LINE 10
- fa si' che il programma parta automaticamente dalla linea 10 quando viene caricato.

Si noti che con il comando SAVE occorre sempre utilizzare un nome di file che non sia gia' contenuto nel catalogo della cartuccia.

VERIFICA DI UN FILE

L'operazione di verifica di un file si svolge esattamente come con le cassette e serve a controllare che la versione memorizzata sia uguale a quella contenuta in memoria. Per esempio:

VERIFY*"m";1;"programma uno"
- confronta il programma in memoria e le sue variabili con quanto contenuto nel file programma uno.

CARICAMENTO E FUSIONE DI FILE

I file possono essere caricati con il comando LOAD oppure con il comando MERGE. Per esempio:

LOAD*"m";1;"prog uno"
- carica in memoria il file "prog uno" dalla cartuccia inserita nel drive 1 cancellando i contenuti precedenti della memoria

LOAD*"m";2;"schermo"SCREEN$
- carica il file "schermo", che e' un blocco di dati binari, nelle opportune locazioni di memoria.
MERGE*"m";3;"prog tre"
- aggiunge al programma in memoria il programma "prog tre", contenuto nella cartuccia inserita nel Microdrive 3.

Si noti che un programma Basic memorizzato con l'opzione LINE viene eseguito automaticamente dopo il caricamento e che un programma cosi' memorizzato non puo' essere caricato con MERGE. Si noti anche che la pressione del tasto BREAK durante il caricamento causa una riinizializzazione del sistema (NEW).

GESTIONE DEI FILE DI DATI

A differenza delle cassette, che permettono soltanto l'uso dei file-programmi appena spiegati, il Microdrive permette di gestire dei veri e propri file sequenziali di dati.

I file sequenziali devono essere sempre letti e scritti record per record a partire dal primo, anche se si puo' creare un'illusione di accesso casuale caricando il file completamente in memoria, per potervi accedere in modo diretto e poi rimemorizzarlo completamente se si sono effettuate delle modifiche.

COME CREARE UN FILE SEQUENZIALE

La creazione di un file sequenziale richiede l'uso dei comandi OPEN, PRINT e CLOSE.

Il comando OPEN e' il primo che deve essere usato nella forma:

10 OPEN#4;"m";1;"...nome..."
- d'ora in poi i dati scritti sul flusso #4 sono destinati al file di nome "...nome..." contenuto nel Microdrive 1.

Durante l'esecuzione del comando OPEN dell'esempio viene creato un canale di Mirodrive nell'area dei dati di

canale. Questo canale viene associato al flusso 4. La cosa piu' importante del canale e' il buffer dei dati, che occupa 512 locazioni e puo' quindi contenere 512 byte di dati che possono essere immagazzinati prima del trasferimento su nastro.

Il buffer dei dati viene riempito con comandi PRINT. Per esempio:

```
20 FOR A=1 TO 300
30 PRINT #4;A
40 NEXT A
```

Dopo che gli elementi dei file sono stati stampati (su file), occorre chiudere il flusso. Per esempio:

```
50 CLOSE #4
```

Il file sul Microdrive viene creato fisicamente soltanto quando il buffer dei dati del canale viene trasferito su nastro la prima volta. Questo succede quando il buffer dei dati viene riempito completamente per la prima volta o quando, con il buffer dei dati parzialmente riempito viene eseguito il comando CLOSE; come nel caso dei file-programmi verra' creato un nuovo file solo se e' stato impiegato un nome di file non ancora utilizzato.

Il contenuto di ogni buffer di dati inviato al file viene sistemato in un settore libero della cartuccia e costituisce un nuovo settore del file. Il descrittore del record, cioe' "l'intestazione" del record, contiene il nome del file e il numero progressivo del buffer inviato. I dati inviati con l'esecuzione del comando CLOSE sono l'ultimo record di un file e recano l'indicatore "end of file" (fine di file). Il comando CLOSE causa anche la chiusura del canale.

LETTURA DI UN FILE SEQUENZIALE

La lettura di un file sequenziale richiede l'uso di OPEN#, INPUT# oppure INKEY$# seguiti da CLOSE#.

Come per la creazione di un file sequenziale, deve essere usato in primo luogo il comando OPEN. Per esempio:

```
10 OPEN#4;"m";1;"...nome..."
```

- questo comando crea un canale di Microdrive associato al flusso specificato.

La prima volta che vengono usati i comandi INPUT e INKEY$, il primo record del file specificato viene trasferito nel buffer dei dati e i record successivi vengono trasferiti quando e' necessario, fino a quando non viene incontrato l'indicatore di fine file.

Un comando del tipo:

```
20 INPUT #4;A
```

legge i byte in ordine dal buffer dei dati fino a quando non incontra un carattere di "ritorno a capo" (ENTER). Il dato letto viene assegnato alla variabile specificata. Si noti la somiglianza col modo in cui l'utente risponde ad un normale comando INPUT premendo i tasti sulla tastiera e terminando con ENTER.

Le stringhe di caratteri possono essere assegnate ad una variabile stringa con comandi del tipo:

```
30 INPUT #4;A$
```

Come per le variabili numeriche tutti i caratteri prima del carattere di ritorno a capo (ENTER) vengono assegnati alla variabile. L'uso di INKEY$ differisce solo nel fatto che viene ritornato un solo carattere ogni volta che INKEY$ viene utilizzato. Per esempio:

```
40 PRINT INKEY$#4;
```

- produce la stampa di un solo carattere letto dal file.

L'uso del comando CLOSE fa si che il canale dei byte venga chiuso e questo produce quindi la perdita di qualunque dato non letto. Si noti che il messaggio "end of file" appare solo nel caso che si cerchi di leggere dei byte che non esistono nel file, cioe' dei byte superato l'ultimo byte del file. Le note del seguente programma spiegano le fasi della creazione e della lettura di un file sequenziale.

Creazione del file:

```
  10 OPEN #12;"m";1;"alfabeto":
REM crea un canale di microdrive
  20 FOR a=65 TO 90
  30 PRINT #12;a: REM scrive 26
numeri nel buffer dei dati
  40 NEXT a
  50 CLOSE #12
```

Lettura del file:

```
  60 OPEN #14;"m";1;"alfabeto":
REM crea un canale di microdrive
  70 FOR b=1 TO 1000: REM legge
il file ad esaurimento
  80 INPUT #14;c
  90 PRINT CHR$ c;: REM stampa i
 caratteri ASCII
 100 NEXT b
```

e, dopo il messaggio "end of file":

CLEAR# oppure CLOSE#14

per chiudere i flussi rimasti aperti.

In questi due programmi il file sequenziale "Alfabeto" contiene un insieme di numeri scritti cifra per cifra, cioe' il numero 10 occupa tre byte, un 1, uno 0 e un ritorno di carrello (ENTER) ma un file sequenziale per contenere l'alfabeto potrebbe essere formato solo da 26 caratteri singoli o una sola stringa di 26 caratteri (che ovviamente occupa meno spazio, perche' si risparmiano 25 ENTER).

MOVE CON FILE SEQUENZIALI

Con il Microdrive non e' possibile trasferire con MOVE i file-programmi cioe' quelli creati con il comando SAVE. Il comando MOVE serve invece per trasferire file di dati, funzione utilissima dato che rende elementare creare delle copie di tali file. Ad esempio per fare una copia del file "Alfabeto" in un nuovo file di nome "Lettere" basta scrivere:

MOVE"m";1;"Alfabeto" TO "m";1;"Lettere"

Il comando MOVE puo' anche essere usato per trasferire dei byte da un file di dati sul Microdrive ad un'altra periferica. Si potrebbero elencare molti esempi, ma i piu' utili sono senz'altro:

MOVE"m";1;"Alfabeto" TO#2
MOVE"m";1;"Alfabeto" TO#3

che consentono di ottenere la stampa sullo schermo o sulla stampante del contenuto di un file sequenziale.

DETTAGLI TECNICI DEL MICRODRIVE

Il Microdrive viene ora discusso sotto tre punti di vista:

- il formato dei dati su nastro
- i canali del Microdrive
- le routine di controllo Basic

Non verra' comunque fatta alcuna discussione sull'elettronica del sistema ne si fara' menzione, per ora, di come utilizzare il linguaggio macchina.

IL FORMATO DEI DATI SU NASTRO

Tutti i dati memorizzati su Microdrive occupano due tracce di nastro, sulle quali i bit di dati vengono memorizzati alternativamente. Questo sistema a due tracce fa si' che, ad una certa velocita', un bit puo' occupare una parte di nastro piu' lungo che se si fosse impiegata una sola traccia per tutti i bit. Ovviamente un Microdrive ha una testina a due tracce indipendenti, cosicche' i dati devono essere divisi in due flussi; la complessita' del sistema e' pero' giustificata dal notevolissimo aumento della velocita' operativa risultante. Lo schema 1 mostra come potrebbero essere immaginati i bit di dati su nastro.

Comunque conviene d'ora in poi dimenticarsi che vengono utilizzate due tracce distinte e pensare che i byte di dati vengano memorizzati su una sola traccia in modo seriale, come mostrato nello schema 2.

Ogni blocco di dati comincia con un'intestazione di 12 byte. Questa intestazione e' formata da 10 byte che contengono 0 e due byte che contengono 255 e consente all'hardware del sistema di identificare l'inizio del blocco di dati con grande precisione.

L'operazione di formattazione di una cartuccia divide il nastro in settori, ognuno dei quali e' costituito dalle seguenti parti:

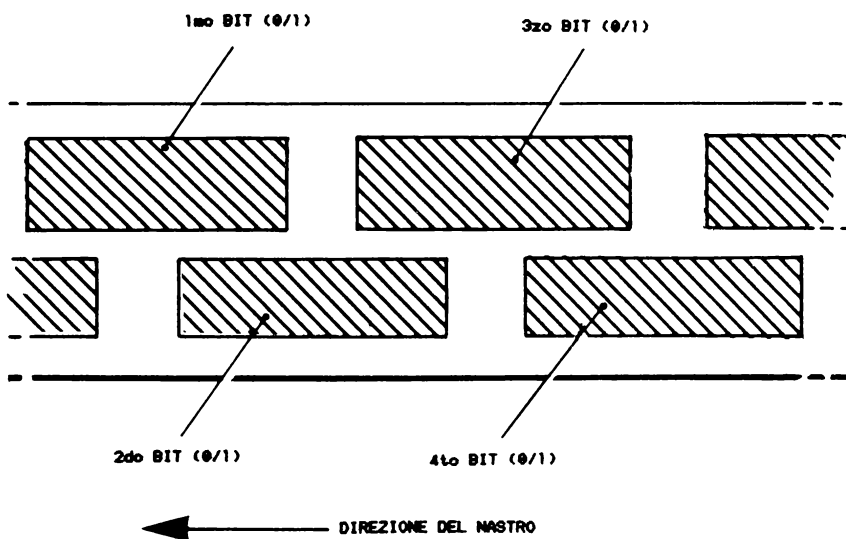



FIG.1. MEMORIZZAZIONE DEI BIT DI DATI SUL NASTRO DEL MICRODRIVE

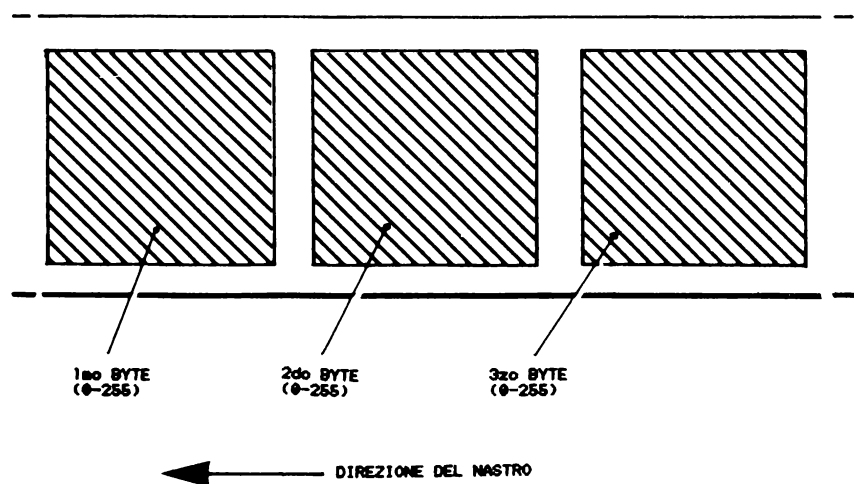



FIG.2. MEMORIZZAZIONE DEI BYTE SUL NASTRO DEL MICRODRIVE
(NOTA: LA LUNGHEZZA E' STATA COMPRESSA DI CIRCA 10 VOLTE RISPETTO ALLA FIG.1)

- Un blocco di intestazione che contiene:

  a) Dodici byte di preambolo

  b) Un'intestazione del blocco lunga 15 byte che contiene:
     - Un byte di flag
     - Un byte con il numero di settore
     - Due byte inutilizzati
     - Dieci byte per il nome della cartuccia
     - Un byte di controllo (checksum)

- Una prima zona libera (gap);

- Un blocco di dati che contiene:

  a) Dodici byte di preambolo
  b) Un descrittore del record lungo 15 byte che contiene:
     - Un byte di flag
     - Un byte con il numero del record
     - Due byte per la lunghezza del record
     - Dieci byte per il nome del file
     - Un byte di controllo (checksum)

  c) Un record formato da:
     - Un'area di dati di 512 byte
     - Un byte di controllo (checksum)

- Una seconda zona bianca (gap).

Per avere un'idea approssimativa delle dimensioni di una parte rispetto all'altra si considerino i seguenti tempi di lettura:

- blocco di intestazione - 1,25 ms.
- prima zona bianca - 3,75 ms.
- blocco di dati - 25 ms.
- seconda zona bianca - 7 ms.

Lo schema 3 mostra i settori su un nastro di Microdrive. Lo schema illustra anche come, nel funzionamento standard, solo due terzi del nastro contengono effettivamente dati.

Lo schema 4 mostra come possono distribuirsi i file nei vari settori.

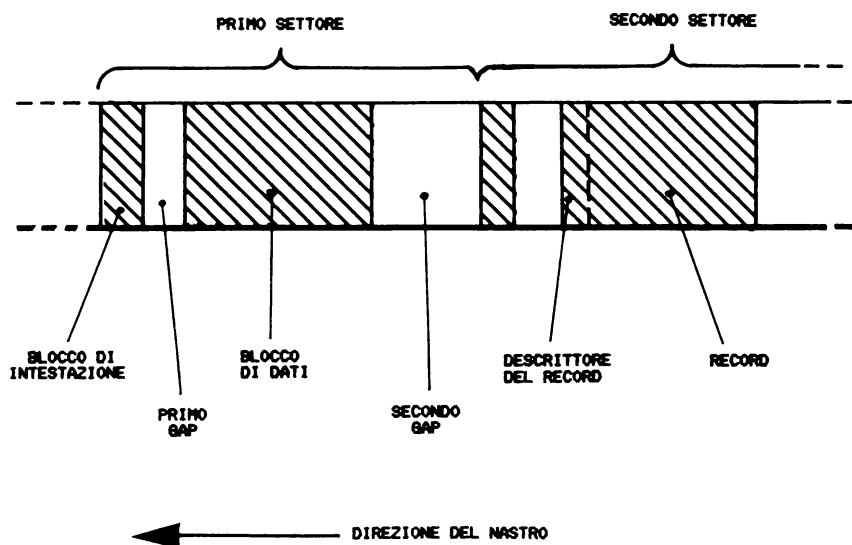
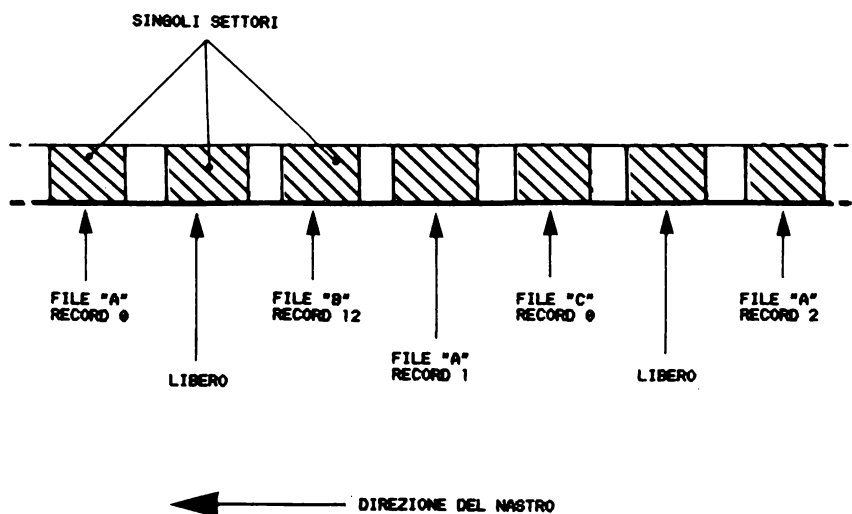



FIG.3. SETTORI SU NASTRO DEL MICRODRIVE (I PREAMBOLI NON SONO INDICATI)

A) FIG.4. FILE SUL NASTRO DEL MICRODRIVE

CANALI DI MICRODRIVE

Tutte le comunicazioni di dati tra lo Spectrum e i Microdrive avvengono attraverso un "canale di Microdrive". La parte piu' importante del canale e' il buffer di dati di 512 byte.

L'utente richiede la creazione di un canale con il comando OPEN, e con esso associa un flusso al canale. In altre circostanze un canale di Microdrive viene creato automaticamente ("ad hoc"). Per esempio, quando viene memorizzato un programma viene creato un canale per trasferire i necessari byte di dati dallo Spectrum alla cartuccia del Microdrive.

Un canale di Microdrive ha il seguente formato:

```
BYTE        CONTENUTO
0-1         Indirizzo 0008h
2-3         Indirizzo 0008h
4           "M"               ("M" + 80h se creato "ad hoc")
5-6         Indirizzo MWRCH (ind. subroutine output)
7-8         Indirizzo MRDCH (ind. subroutine input)
9-10        Numero '595'      (lunghezza canale Microdrive)
11-12       CHBYTE            (contatore per l'area dei dati)
13          CHREC             (numero buffer - comincia da 0)
14-23       CHNAME            (nome del file)
24          CHFLAG            (flag di lettura/scrittura)
25          CHDRIV            (num. unita' Microdrive)
26-27       CHMAP             (indirizzo della mappa
                              corrente)
;   le seguenti 27 locazioni sono l'area di lavoro
;   del blocco di intestazione:
28-39       Intestazione del blocco di intestazione
40          HDFLAG            (byte di flag)
41          HDNUMB            (numero settore)
42-43       inutilizzato
44-53       HDNAME            (nome della cartuccia)
54          HDCHK             (byte di controllo per gli
                              ultimi 14 byte = checksum)
;   le seguenti 540 locazioni sono l'area di lavoro
;   blocco dei dati:
55-56       Intestazione del blocco dei dati
67          RECFLG            (byte di flag)
68          RECNUM            (numero buffer)
69-70       RECLEN            (lunghezza del buffer attuale)
```

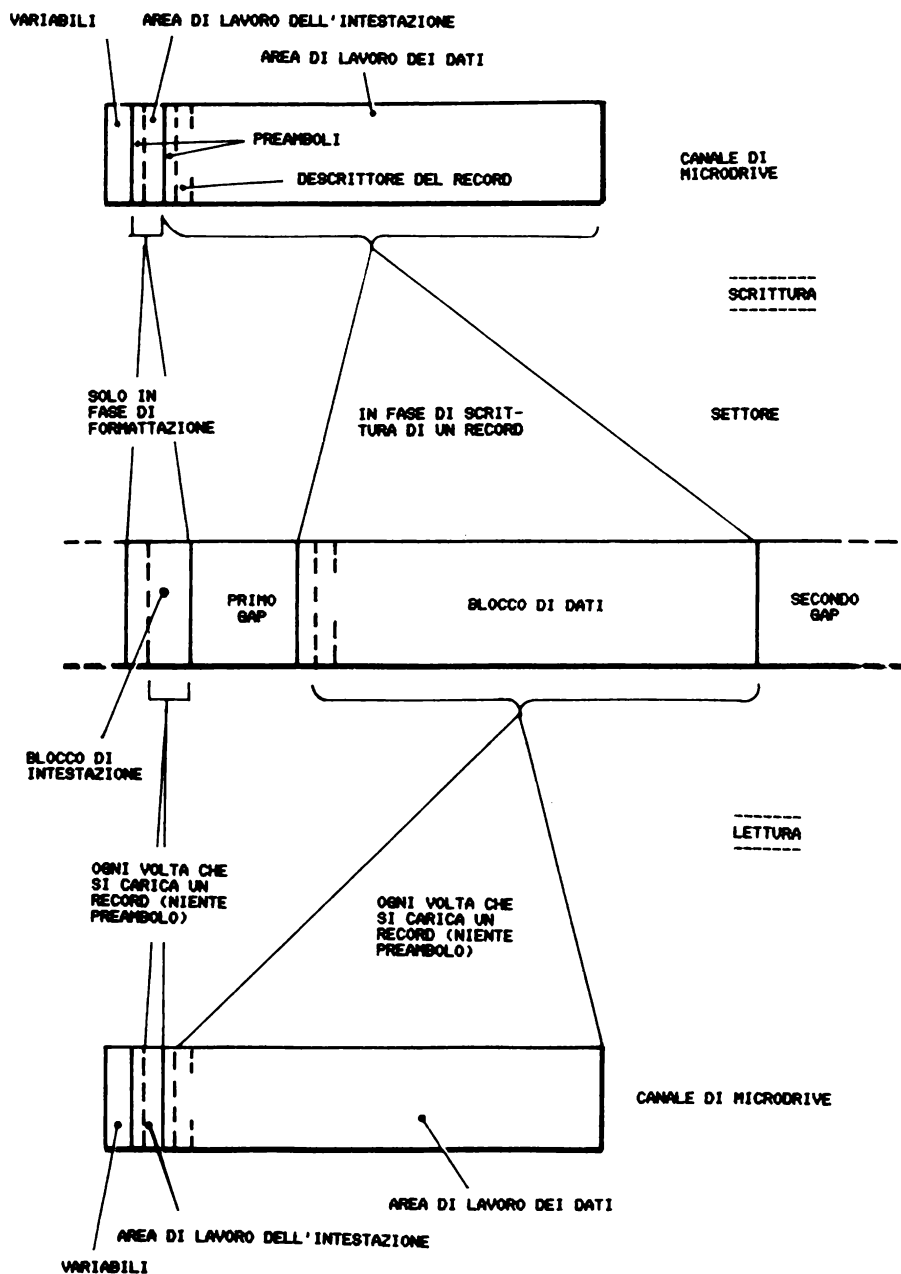



FIG.5. LETTURA E SCRITTURA DI RECORD

71-80	RECNAM	(nome del file)
81	DESCHK	(byte di controllo per gli ultimi 14 byte)
82-593	L'area di dati di 512 locazioni	
594	DCHK	(byte di controllo per l'area dei dati)

Viene ora preso in esame il modo in cui vengono usate le locazioni del canale, considerando le routine dei vari comandi Basic che usano il Microdrive. Si consiglia di fare riferimento allo schema 5 che illustra la lettura e la scrittura di un record.

ROUTINE DI CONTROLLO DEI COMANDI BASIC

In questa sezione non viene discussa la codifica delle routine ma soltanto lo schema generale delle funzioni che svolgono. Si vuole fornire cosi' una visione chiara del funzionamento del Microdrive.

F O R M A T

L'operazione di formattazione di una cartuccia deve essere ovviamente la prima ad essere discussa, anche se e' piuttosto complicata. E' suddivisa nelle seguenti operazioni:

i. Viene creato un canale "ad hoc" per la comunicazione tra lo Spectrum e il Microdrive. Il numero di Microdrive viene sistemato nella variabile di sistema CHDRIV e il nome della cartuccia in CHNAME.

ii. Viene preparata una mappa di Microdrive nella zona delle mappe, cioe' tra le variabili di sistema e l'area dei dati di canale.

- Una mappa di Microdrive occupa 32 byte ed e' utilizzata per contenere 256 flag (8 X 32 = 256) che indicano quali settori di una cartuccia sono liberi per essere usati.

Tutti i bit della mappa sono inizializzati ad 1 = in uso.

iii. Viene avviato il motore del Microdrive il cui numero e' indicato dalla variabile CHDRIV.

iv. Sono inizializzati i byte dell'intestazione dell'area di lavoro.

v. Sono inizializzati i 540 byte del blocco di dati (l'area dei dati contiene ora dati per eseguire un controllo).

vi. Ora possono essere creati i settori sul nastro. Il blocco di intestazione viene copiato dall'area di lavoro dell'intestazione (27 byte) e il blocco dei dati viene copiato dall'area di lavoro dei dati (540 byte). In totale 255 settori sono creati con in genere gli ultimi 60 che vanno a riscriversi sopra i primi, perche' il nastro e' finito e ricomincia da capo. I settori vengono numerati da 255 in ordine decrescente.

vii. I dati di controllo vengono ora verificati leggendo tutti i record. Se il controllo di un record rivela che questo funziona correttamente allora viene resettato il bit relativo nella mappa del Microdrive, cio' indica che quel settore puo' essere impiegato.

viii. Viene ora eseguita una seconda operazione di scrittura con RECFLAG e RECLEN a 0. La scrittura viene eseguita su tutti i settori impiegabili (funzionanti) e in questo modo tutti questi settori diventano disponibili per i file.

ix. Viene spento il motore del Microdrive.

x. Viene chiuso il canale del Microdrive.

C A T

La stampa della lista dei file contenuti in una cartuccia e' suddivisa nelle seguenti operazioni:

i. Viene reso corrente il flusso richiesto per la stampa (normalmente viene utilizzato il flusso #2).

ii. Viene creato "ad hoc" un canale di Microdrive

iii. Viene preparata una mappa di Microdrive. Come con il comando FORMAT i 256 bit sono tutti settati per indicare che l'operazione riguarda tutti i settori.

iv. Viene avviato il motore del Microdrive indicato da CHDRIV.

v. Vengono esaminati uno dopo l'altro tutti i settori della cartuccia.
Se un settore non contiene alcun file, viene resettato il bit relativo nella mappa del Microdrive, altrimenti il nome del file contenuto in quel settore, viene inserito in una lista di nomi di file costituita nell'area dei dati. Vengono inseriti soltanto i nomi di file che non fanno gia' parte della lista e vengono trascurati tutti i nomi che cominciano con CHR$ 0. Ogni volta che viene trovato un nuovo file, la lista viene riordinata in ordine alfabetico.
Il nome della cartuccia contenuto in ogni blocco di intestazione del settore, viene caricato nell'area di lavoro dell'intestazione del canale.

vi. I risultati delle precedenti operazioni sono stampati in questo ordine:

- il nome della cartuccia contenuto nella variabile HDNAME.
- i vari nomi di file, contenuti nell'area dei dati.
- la quantita' di memoria libera, determinata esaminando la mappa, cioe' contando la quantita' di bit resettati e dividendo il risultato per due.

vii. Vengono chiusi il canale di Microdrive e la mappa.

E R A S E

L'operazione di cancellazione di un file da una cartuccia e' suddivisa nelle seguenti operazioni:

i. Viene creato ad hoc un canale di Microdrive e una mappa.

ii. Viene avviato il motore del Microdrive, indicato da CHDRIV.

iii. Vengono resettate le prime 32 locazioni dell'area di dati. Questi byte serviranno come mappa per indicare quali settori dovranno essere cancellati, cioe' resi disponibili.

iv. Viene inizializzato un contatore a 1280; verra' decrementato per contare i settori (1280 settori = almeno cinque passaggi di nastro).

v. Vengono esaminati uno dopo l'altro tutti i descrittori di record di tutti i settori del nastro; se un settore contiene un record del file da cancellare, viene settato il relativo bit nella mappa appositamente creata, altrimenti il settore viene ignorato .

Quando viene trovato il settore che contiene il record con la segnalazione di fine file il numero di tale record viene caricato nella locazione CHREC.

L'analisi dei settori procede fino a quando o il contatore dei settori raggiunge 0 o il numero di record esaminati diventa uguale al numero caricato in CHREC. Ecco perche', se non si trova l'indicazione di fine file, l'operazione richiede circa 40 sec. (5 passaggi del nastro).

vi. A questo punto e' necessaria un'ultima operazione di scrittura. Vengono scritti in tutti i settori destinati alla cancellazione i descrittori di record libero.

vii. Viene fermato il motore del Microdrive.

viii. Vengono chiusi il canale e la mappa del Microdrive.

S A V E

Il comando SAVE e' il primo dei comandi che scrivono dati su una caruccia. Deve essere creata una mappa di Microdrive per la cartuccia in uso, che indichi i settori utilizzabili. Inoltre occorre confrontare il nome del nuovo file con i nomi dei file gia' memorizzati per assicurarsi che non sia gia' stato impiegato. L'esecuzione di questo comando e' suddivisa nelle seguenti operazioni:

i. Viene creato "ad hoc" un canale e una mappa di Microdrive.

ii. Viene avviato il motore del Microdrive, il cui numero e' contenuto in CHDRIV.

iii. Tutti i settori della catuccia vengono esaminati in sequenza. Viene costruita una mappa di Microdrive che indica quali settori sono disponibili per l'uso; vengono esaminati tutti i nomi di file contenuti, per assicurarsi che il nuovo nome di file puo' essere utilizzato.

iv. Viene scritta nell'area di dati del canale del Microdrive un'intestazione di 9 byte che descrive il programma da memorizzare.

Tale intestazione descrive il programma in modo simile a quella usata con le cassette:

```
byte 1   - byte di codice
           0   = programma Basic
           1/2 = matrici
           3   = codice binario
byte 2&3 - lunghezza del blocco
byte 4&5 - indirizzo del blocco
byte 6&7 - lunghezza del solo programma
byte 8&9 - il numero di linea se viene
           utilizzata l'opzione LINE.
```

v. Vengono trasferiti i byte del programma nell'area dei dati del Microdrive, ma quando tale area e' piena, cioe' quando dovrebbe essere inviato il 513mo byte, essa deve essere trasferita sulla cartuccia. Questa operazione richiede:

- Trovare il blocco di intestazione del primo settore successivo sul nastro.
- Esaminare la mappa del Microdrive per vedere se tale settore e' libero; se non lo e' occorre attendere il settore successivo.
- Copiare i 540 byte dall'area di lavoro del blocco dei dati sul nastro, creando cosi' un nuovo record.
- Settare il bit relativo al settore nella mappa del Microdrive, indicando cosi' che il settore contiene dati in uso.

Il valore di RECNUM, che contiene il numero progressivo dei settori, viene incrementato ogni volta che viene eseguita questa operazione.

vi. Viene creato un record di fine file. Questo richiede che venga settato il flag RECFLAG che vengano trasferiti i 540 byte dall'area di lavoro del blocco di dati sul nastro un'ultima volta.

vii. Viene fermato il motore del Microdrive.

viii. Viene chiuso il canale del Microdrive.

L O A D , V E R I F Y & M E R G E

Le routine di questi tre comandi sono identiche dal punto di vista del software del Microdrive e possono quindi essere considerate insieme. Sono suddivise nelle seguenti operazioni:

i. Viene creato ad hoc un canale e una mappa di Microdrive.

ii. Viene avviato il motore del Microdrive il cui numero e' contenuto in CHDRIV.

iii. Viene caricato nell'area di lavoro il blocco di dati del settore che contiene il primo record del programma.

iv. I primi 9 byte del programma che contengono l'intestazione vengono opportunamente interpretati a seconda del comando in uso e del tipo di programma. Il contenuto del primo buffer e' quindi caricato (LOAD) oppure confrontato con il contenuto della memoria (VERIFY), oppure aggiunto al programma in memoria (MERGE).

v. Gli altri settori contenenti i vari record del programma vengono caricati nel giusto ordine, fino a quando non viene incontrato il record con l'indicazione di fine file.

vi. Viene fermato il motore del Microdrive.

vii. Vengono chiusi il canale e la mappa del Microdrive.

O P E N

Molte delle operazioni richieste da un comando OPEN sono diverse a seconda che si voglia leggere un file gia' esistente o creare un nuovo file. Si ricordi che con i Microdrive dello Spectrum non e' possibile aggiungere dati ad un file esistente. Questo comando e' suddiviso nelle seguenti operazioni:

i. Viene creato un canale di Microdrive che viene associato al flusso specificato da S-STR1, al Microdrive il cui numero e' specificato in D-STR1 e al nome del file contenuto in N-STR1.

ii. Viene avviato il motore del Microdrive indicato da CHDRIV.

iii. Tutti i settori del nastro vengono esaminati in sequenza. Viene costruita una mappa del Microdrive che indica quali sono i settori liberi. Questa verra' utilizzata nel caso della creazione di un nuovo file. Infine tutti i nomi di file contenuti nella cartuccia vengono confrontati con il nome di file specificato.

iv. Per la scrittura di un file: se non vengono incontrati settori che recano il nome di file richiesto si assume che l'utente desidera creare un file per la scrittura. In questo caso viene eseguito il passo 5.

Per la lettura di un file esistente: nel caso che venga trovato un file con lo stesso nome vengono settati gli opportuni indicatori per la lettura e il primo record del file viene copiato nell'area di lavoro dei dati nel canale del Microdrive.

v. Viene fermato il motore del Microdrive.

P R I N T

Nello Spectrum l'esecuzione del comando PRINT produce l'invio di una serie di caratteri ad una certa periferica che e' quella associata al flusso corrente. Quindi quando l'utente scrive PRINT #5;...(per esempio), i caratteri vengono inviati all'appropriata routine di OUTPUT. Nel

caso di un flusso di Microdrive la routine ha nome MWRCH (= Microdrive write character, scrittura di un carattere sul Microdrive).

L'effetto della routine MWRCH e' di aggiungere un carattere al buffer del canale di Microdrive in uso; ma ogni volta che il buffer viene riempito completamente i suoi contenuti sono trasferiti sul primo settore libero della cartuccia, formando un nuovo record del file. Il contatore del buffer (e quindi il numero del record) viene incrementato ogni volta che viene trasferito un buffer. Si noti che con l'uso di comandi PRINT non puo' essere creato un indicatore di fine file.

I N K E Y $

L'effetto della funzione INKEY$ e' diverso rispetto al comando PRINT.

Ogni volta che occorre leggere un codice di carattere viene chiamata la routine MRDCH (Microdrive read character, lettura di un carattere dal Microdrive). Questa routine ritorna un solo codice di un carattere leggendolo dall'appropriato canale di Microdrive; quando il buffer e' vuoto viene caricato un nuovo settore del file dalla cartuccia inserita nel Microdrive specificato. Nel caso che il buffer sia vuoto e l'indicatore di fine di file sia gia' settato viene visualizzato il messaggio di "end of file".

I N P U T

L'esecuzione di un comando INPUT viene effettuata usando piu' volte consecutivamente la routine MRDCH. I caratteri vengono caricati uno dopo l'altro nell'area di edit fino a quando non viene incontrato un carattere di ritorno di carrello (= ENTER); quindi i caratteri letti vengono assegnati alla variabile specificata.

C L O S E

La chiusura di un canale di Microdrive e' suddivisa nelle seguenti operazioni:

1. Viene esaminato il valore contenuto nella variabile CHFLAG associata al flusso. Se il valore e' 0 allora il canale e' usato per la lettura e occorre semplicemente chiudere il canale perdendo ogni dato contenuto nel buffer. Diversamente, il canale e' usato per la scrittura e vengono eseguite anche le operazioni illustrate di seguito.

2. L'opportuno bit di RECFLAG viene settato per indicare che sta per essere creato un record che chiude il file.

3. Viene avviato il motore del Mirodrive, indicato da CHDRIV.

4. Viene esaminata ogni intestazione del settore fino a quando non viene identificato un settore libero (riferendosi alla mappa del canale).

5. Viene scritto nel settore un record che reca l'indicazione di fine di file.

6. Viene fermato il motore del Microdrive.

7. Vengono chiusi il canale e la mappa del Microdrive.

M O V E

La routine per questo comando e' piuttosto complicata dato che deve tener conto di tutte le diverse periferiche che possono essere utilizzate. Comunque alla base dell'operazione di MOVE c'e' la ripetizione di una routine di lettura seguita da una routine di scrittura; il programma seguente ne fornisce un esempio.

Il comando MOVE "m";1;"a" TO "m";1;"b" e' equivalente a:

```
10 OPEN #4;"m";1;"a"
20 OPEN #5;"m";1;"b"
30 PRINT #5;INKEY$#4: GO TO 30
```

ma l'esecuzione termina con il messaggio 'end of file' (fine di file) dato che il programma non conosce la lunghezza del file.

UNA NOTA SULLE COPIE

E' possibile usando, per esempio, "POKE 23791,20" prima di un comando SAVE, fare piu' copie di ogni record del programma memorizzato. Questa operazione naturalmente provoca l'occupazione di molto piu' spazio sulla cartuccia, ma permette un notevole risparmio di tempo quando il programma viene caricato.

COME ESAMINARE RECORD E SETTORI

I seguenti due programmi Basic indicano come sia possibile trovare in un file sequenziale i numeri dei settori usati per i record del file.

```
10 OPEN #4;"m";1;"a"
20 FOR a=1 TO 3000
30 PRINT #4;CHR$ 65;
40 NEXT a
50 CLOSE #4
```

queste linee Basic creano un file di nome "A" che contiene 3000 lettere A.
Il file A puo' essere letto usando:

```
NEW
  10 OPEN #4;"m";1;"a"
  20 PRINT "record ";PEEK 23912,
"settore ";PEEK 23885
  30 POKE 23856,3
  40 LET a$=INKEY$#4
  50 GO TO 20
```

NOTE:

- Il NEW iniziale fa si che il canale del Microdrive si trovi dalla locazione 23844 in su.

- Il numero di record viene letto dall'area di lavoro dei dati e il numero di settore dall'area di lavoro dell'intestazione.
- Il POKE 23856,3 inserisce un valore alto in CHBYTE e ha l'effetto di svuotare il buffer dei dati.

Sul calcolatore dell'autore si sono ottenuti i seguenti risultati per il file "a":

```
record 0        settore 27
record 1        settore 21
record 2        settore 15
record 3        settore 9
record 4        settore 3
record 5        settore 180

8 End of file, 40:1
```

COME ESAMINARE LA MAPPA DEL MICRODRIVE

Fino a questo momento e' stato detto poco a proposito della mappa del Microdrive, anche perche' non e' mai necessario alterarne i contenuti. Esaminare una mappa e' comunque interessante; i seguenti programmi Basic illustrano come si puo' fare.

- Iniziate formattando una cartuccia, per esempio:
  FORMAT "m",1,"mappa"
- usate NEW e ENTER per cancellare ogni mappa o canale esistenti in memoria
- e scrivete:

```
  10 OPEN #4;"m";1;"nuovo": REM
nuovo file
  20 FOR a=23792 TO 23823: REM m
appa per "nuovo"
  30 LET n=PEEK a: REM ogni byte
  40 FOR b=1 TO 8: REM 8 bit per
 byte
  50 PRINT n/2<>INT (n/2);: REM
"0" o "1" per ogni bit
  60 LET n=INT (n/2)
  70 NEXT b
  80 NEXT a
  90 CLEAR #
RUN
```

L'autore con una cartuccia nuova ha ottenuto i seguenti risultati:

```
10000000000000000000000000000000
00000000000000000000000000000000
00000000000000000110000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000001111111111
11111111111111111111111111111111
11111111111111111111111111111111
```

I bit di settore sono dati in ordine crescente, settore 0, settore 1, ..., settore 256, per quanto sul nastro vengano incontrati in ordine decrescente. In questa mappa il settore 0 non esiste, i settori 81 e 82 sono non utilizzabili, probabilmente sono il punto dove il nastro e' giuntato, e i settori 182-255 non esistono. Ci sono quindi 179 settori che possono essere impiegati, ecco perche' CAT 1 e ENTER indica che su questa cartuccia sono disponibili 90K di memoria.

Dopo aver memorizzato un programma Basic di 6.5 K la mappa appariva cosi':

```
10000000000000000000000000000000
00000000000000000000000000000000
00000000000000000110000000000010
10101010101010101010101000000000
00000000000000000000000000000000
00000000000000000000001111111111
11111111111111111111111111111111
11111111111111111111111111111111
```

Si noti che i record non sono in settori adiacenti, infatti dopo la creazione di un record non c'e' abbastanza tempo per caricare altri 512 byte nell'area di canale prima del passaggio del record successivo.

NOTA CONCLUSIVA - IOBORD

La variabile di sistema fantasma IOBORD, locazione 23750 (5CC6h), contiene normalmente 0 e quindi il bordo dello schermo televisivo lampeggia in nero quando vengono trasferiti dei blocchi di dati dallo Spectrum al Microdrive (ma non viceversa). L'utente puo' variare il valore di IOBORD nell'intervallo 0-7 per cambiare a suo piacimento il colore del bordo dello schermo durante la scrittura su nastro.

FOTO 1. IL SISTEMA SPECTRUM ESTESO (L'INTERFACCIA ZX1 NON E' VISIBILE)

FOTO 2. VISTA POSTERIORE DELL'INTERFACCIA ZX1

FOTO 3. VISTA INTERNA DELL'INTERFACCIA ZX1

FOTO 4. CARTUCCIA CON CUSTODIA

ULA DEL MICRODRIVE
SULLA SCHEDA
SUPERIORE

MOTORE

CONNETTORE
TERMINALE
(PER SPECTRUM)

CONNETTORE
TERMINALE

SCHEDA C.S.
INFERIORE

CONNETTORE
MICRODRIVE-MICRODRIVE

CAVO
PIATTO

CONNESSIONE
TRA C.S.

FOTO 5. VISTA POSTERIORE DI MICRODRIVE APERTO

C A P I T O L O   4

L A   R E T E   L O C A L E

NOTA DELL'AUTORE

Per la rete non esistono informazioni riservate e quindi essa puo' essere esaminata in dettaglio; e' importante permettere la diffusione dello standard Sinclair per collegare un calcolatore ad un altro.

INTRODUZIONE

La rete locale dello Spectrum, che d'ora in poi chiameremo semplicemente rete, rende possibile collegare un certo numero di Spectrum (o di calcolatori di altre marche) in modo da poter trasmettere dati con grande velocita' da una stazione all'altra. I dati vengono trasmessi sulla rete ad una velocta' superiore a quella consentita dall'interfaccia RS232, teoricamente 5 Kbyte per secondo (8 bit per ogni byte).

Per chi non e' familiare con l'uso di una rete, vengono fornite alcune importanti definizioni di carattere generale.

> RETE - Due o piu' calcolatori collegati tra loro con lo scopo di scambiarsi dati.
> STAZIONE - Computer collegato alla rete.
> NUMERO DI STAZIONE - Numero assegnato ad una certa stazione. Di solito una stazione risponde sempre allo stesso numero, ma in alcune circostanze e' utile usare provvisoriamente un numero diverso.
> SORGENTE - Stazione che trasmette i dati.
> DESTINAZIONE - Stazione che riceve i dati.

Le definizioni seguenti valgono in questa trattazione per la rete dello Spectrum.

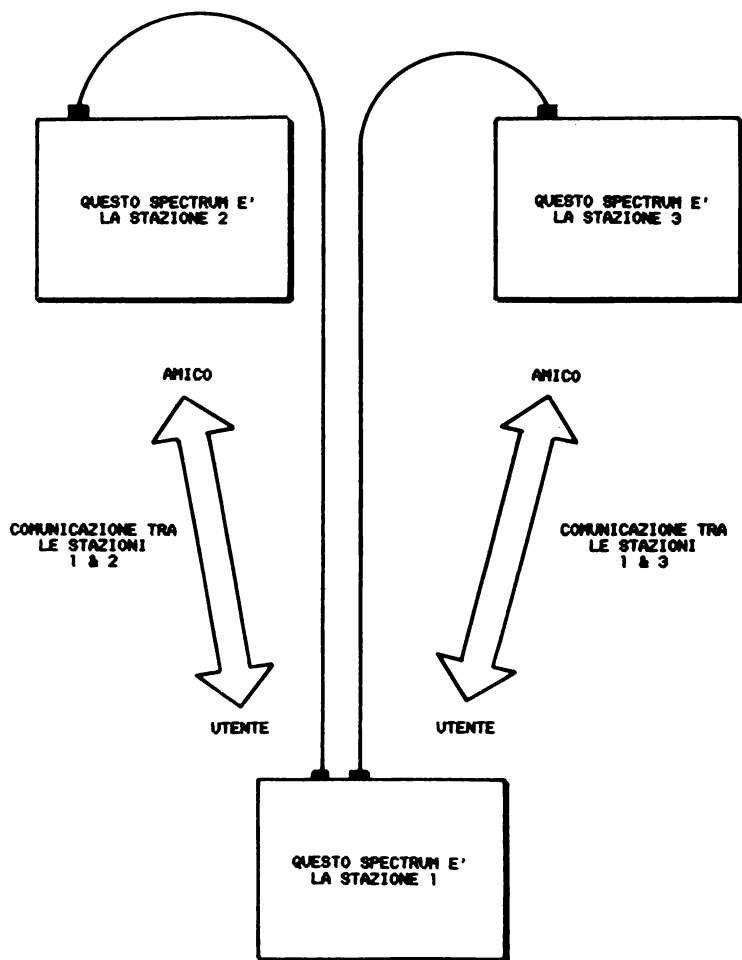
> UTENTE - Nome usato per indicare lo Spectrum dell'utente.

NOTA: GLI OPERATORI DEGLI SPECTRUM 2 & 3 HANNO UNA VISIONE ANALOGA DELLA RETE

FIG.1. RETE FORMATA DA TRE STAZIONI

AMICO - Nome usato per indicare la stazione in comunicazione con UTENTE. Le comunicazioni avvengono sempre da UTENTE verso AMICO oppure da AMICO verso UTENTE.

USO DELLA RETE

Nella maggior parte dei casi, la rete viene sfruttata dal BASIC attraverso dei comandi che vengono ora discussi individualmente dal punto di vista dell'utente.

COME CAMBIARE NUMERO DI STAZIONE

All'accensione uno Spectrum (equipaggiato con interfaccia ZX1) non possiede alcun numero di stazione; lo Spectrum diventa la stazione 1 solo quando la memoria fantasma dell'interfaccia viene inserita per la prima volta nel sistema. Il numero di stazione e' contenuto nella variabile di sistema fantasma NTSTAT, di indirizzo 23749, che puo' essere alterata con:

FORMAT"n", numero di stazione

oppure:

POKE 23749, numero di stazione

Normalmente il numero di stazione deve essere compreso nell'intervallo 1-64, anche se questa restrizione cade usando la rete in linguaggio macchina.

Il numero di stazione viene copiato nell'area delle informazioni di canale ogni volta che viene aperto un nuovo canale della rete. Quindi, se tale numero viene cambiato in relazione all'apertura di nuovi canali, e' possibile, e anche utile, avere aperti contemporaneamente piu' canali con diversi numeri di stazione, magari uno per ricevere e uno per trasmettere dati.

TRASMISSIONE DI PROGRAMMI, MATRICI O DATI BINARI

Programmi Basic, matrici definite o blocchi di dati binari possono essere inviati ad un'altra stazione (da

UTENTE ad AMICO) usando il comando SAVE. Per esempio:

SAVE*"n",2
- dove AMICO e' la stazione 2
SAVE*"N";3 LINE 10
- dove AMICO e' la stazione 3; il programma si lancia da solo.
SAVE*"N";0 SCREEN$
- dove AMICO puo' essere chiunque in attesa sul canale di trasmissione collettiva e i dati da trasmettere sono l'immagine video.

In tutti questi esempi, condizione necessaria affinche' la comunicazione abbia luogo e' che la stazione AMICO sia in attesa dei dati. Se il numero di stazione specificato e' diverso da 0, AMICO deve essere in attesa di una comunicazione da parte di UTENTE; se invece il programma viene inviato sul canale di trasmissione collettiva (numero di stazione = 0) tutte le stazioni che desiderano ricevere il messaggio devono essere in attesa su questo canale.

Lo scambio di dati con una certa stazione, nei nostri esempi AMICO, richiede uno scambio di informazioni preliminari di riconoscimento. Una trasmissione collettiva (NTSTAT = 0), invece, non richiede alcun riconoscimento e puo' quindi essere ricevuta da piu' stazioni contemporaneamente. I dati vengono trasmessi sul canale di trasmissione collettiva ad una velocita' circa quattro volte inferiore a quella usata con gli altri canali.

LOAD, MERGE, VERIFY CON PROGRAMMI, MATRICI E DATI BINARI

Programmi Basic, matrici o dati binari trasmessi da un'altra stazione (comunicazioni da AMICO a UTENTE) possono essere ricevuti con i comandi LOAD, MERGE e VERIFY. Per es.:

LOAD*"n";2 DATA
- la sorgente e' la stazione 2
VERIFY *"n",0
- il programma inviato sul canale di trasmissione collettiva viene confrontato con quello contenuto in memoria.
MERGE*"N";1
- il programma ricevuto dalla stazione 1 viene

fuso col programma in memoria. Si noti che non e' importante se anche UTENTE ha 1 come numero di stazione.

Anche in questo caso e' indispensabile che AMICO introduca il comando di SAVE per trasmettere i dati attesi.

Lo schema 2 indica il protocollo per trasmettere un programma Basic, una matrice o dati binari attraverso la rete. Non e' pero' possibile mostrare in un diagramma lo stato di attesa del calcolatore. Per esempio se SAVE*"N";2 viene introdotto prima di LOAD*"N";1 la stazione 1 aspettera' che la stazione 2 sia pronta e vice versa. La trasmissione ha luogo soltanto quando entrambe le stazioni sono pronte. I dati vengono suddivisi in buffer di 255 byte e, quando una stazione e' in attesa, prima di ricevere o trasmettere ogni buffer, il bordo dello schermo televisivo lampeggia (analogamente ai caricamenti da cassetta). Il colore del bordo sara' alternativamente il colore normale e il colore contenuto nella variabile di sistema fantasma IOBORD di indirizzo 23750 (nell'intervallo 0-7). L'utente puo' cambiare il valore di IOBORD come desidera.

L'uso dei comandi SAVE, LOAD, VERIFY e MERGE non richiede l'apertura o la chiusura di flussi. Infatti nello Spectrum UTENTE viene aperto "ad hoc", cioe' appositamente per lo scopo, un canale della rete che viene immediatamente chiuso appena la funzione di SAVE, LOAD o VERIFY e' completata. La gestione di questi canali e' quindi totalmente trasparente all'utente.

COME TRASMETTERE DATI SULLA RETE

UTENTE puo' trasmettere dati sulla rete usando tre comandi Basic: OPEN#, PRINT# e CLOSE#.

I dati possono essere considerati come:

- un insieme di espressioni stampabili separate da caratteri di ritorno carrello (= ENTER);

oppure:

- un insieme di singoli caratteri.

Il primo passo necessario per poter trasmettere dei dati da UTENTE ad AMICO e' creare un canale della rete. A

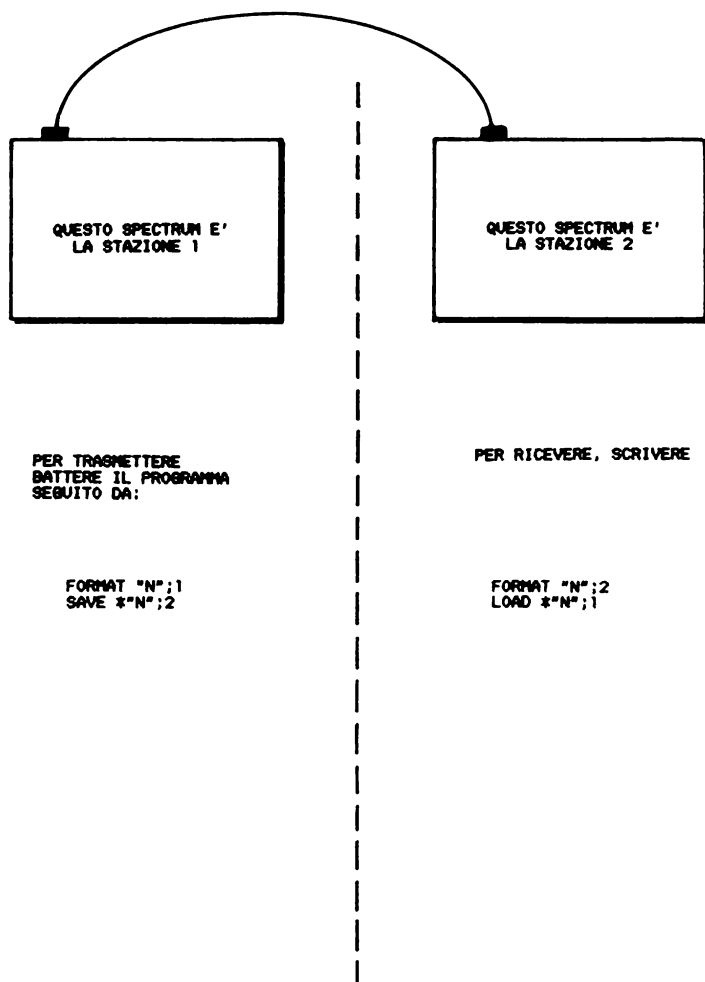
PRIMA:
(a) GLI OPERATORI SI METTONO D'ACCORDO DI USARE I NUMERI DI STAZIONE 1 & 2;
(b) GLI OPERATORI CONCORDANO CHE LA STAZIONE 1 INVIERA' UN PROGRAMMA ALLA STAZIONE 2.

DOPO:
I DUE SPECTRUM CONTENGONO GLI STESSI PROGRAMMI

FIG.2. TRASMISSIONE DI UN PROGRAMMA BASIC ATTRAVERSO LA RETE

questo scopo si usa il comando OPEN#.

OPEN#4;"N";18
- crea un canale della rete per comunicare da UTENTE ad AMICO, la stazione 18, e lo associa al flusso 4.
OPEN#5;"n",0
- crea un canale della rete per la trasmissione collettiva dei dati e lo associa al flusso 5.

Piu' avanti in questo capitolo si esamineranno in dettaglio i contenuti di un canale della rete; per ora basti sapere che contiene sempre un buffer di 255 byte. Questo significa che si possono accumulare 255 caratteri prima che si renda effettivamente necessaria una trasmissione.

Il secondo passo e' di riempire il buffer con i dati che si desiderano trasmettere. A questo scopo si usa il comando PRINT# con lo stesso numero di flusso specificato nel comando OPEN.
Se i dati devono essere considerati espressioni formate da piu' caratteri, da leggere col comando INPUT (vedi dopo), occorre inserire i ritorni di carrello nelle opportune posizioni, cioe' nella posizione corrispondente alla pressione del tasto ENTER, in risposta ad un INPUT da tastiera. Ecco alcuni esempi di comandi PRINT#:

PRINT#4;1
- i caratteri che vengono scritti nel buffer sono: "1" e "ritorno di carrello".
PRINT#4;"UNO"
- i caratteri che vengono scritti nel buffer sono: "U","N","O" e "ritorno di carrello".
PRINT#4;A'B$
- i caratteri che vengono scritti nel buffer sono: i caratteri prodotti dalla stampa della variabile A, "ritorno di carrello", i caratteri prodotti dalla stampa della variabile B$ e "ritorno di carrello".

Il buffer usato con la rete puo' contenere fino a 255 caratteri. Quando si cerca di scrivere il 256mo carattere il buffer, gia' riempito, viene trasmesso verso la stazione AMICO e il 256mo carattere diventa il primo carattere del buffer successivo.
Il terzo passo e' di chiudere il flusso. Questa funzione viene svolta dal comando Basic CLOSE# che invia i contenuti del buffer (solo parzialmente riempito) alla

stazione AMICO, chiude il canale della rete e infine chiude il flusso scrivendo degli 0 al posto dei dati di flusso, oppure i valori originari per i flussi 0-3. Per esempio:

CLOSE#4
- chiude il flusso 4.

COME RICEVERE DATI DALLA RETE

Per ricevere dati dalla rete si usano i comandi Basic OPEN#, INPUT#, INKEY$# e CLOSE#. Come per la trasmissione, i dati possono essere considerati: espressioni separate da caratteri "ENTER", oppure insieme di singoli caratteri.

Il primo passo di questa operazione riguarda anche in questo caso l'apertura di un flusso. Per esempio:

OPEN#7;"N";1
- crea un canale della rete per comunicare da AMICO, stazione 1, a UTENTE, e lo associa al flusso 7.
OPEN#6,"n",0
- crea un canale della rete che puo' essere usato per ricevere una trasmissione collettiva, e lo associa al flusso 6.

Il secondo passo e' usare opportunamente i comandi INPUT# e INKEY$#. Si noti che INPUT# non verra' usato nel caso che si desiderino leggere dei caratteri singoli. Ecco alcuni esempi di comandi di lettura:

INPUT#7;A
- un certo numero di byte ricevuti, fino al ritorno di carrello, sono assegnati alla variabile A. L'espressione che viene assegnata alla variabile deve essere numerica.
INPUT#7;A$
- l'espressione deve essere una stringa di caratteri di lunghezza finita.
INKEY$#7
- questa funzione ritorna una stringa formata dal primo carattere non ancora letto nel buffer.

Con tutti i comandi di questo tipo il computer controlla se ci sono dati validi, cioe' ricevuti ma non ancora letti, nel buffer; se non ne trova carica il buffer successivo da AMICO. Pero' se l'ultimo buffer portava l'indicazione di fine file, viene visualizzato il messaggio di errore di fine file ("end of file").

INKEY$ usato con una determinata stazione (cioe' non in trasmissione collettiva) ha una caratteristica particolare. Usualmente questa funzione ritorna una stringa di lunghezza unitaria formata dal carattere successivo all'ultimo letto dal buffer, ma se AMICO non e' in attesa di una richiesta da parte di UTENTE di trasmettere il sucessivo buffer di dati, INKEY$ ritorna una stringa nulla. Questa caratteristica permette a UTENTE di effettuare un ciclo di lettura (polling) per poter gestire comunicazioni con piu' stazioni contemporaneamente, ricevendo i dati solo quando queste sono pronte a trasmetterli senza dover perdere tempo in attesa.

Il terzo passo e' di chiudere il flusso usando il comando CLOSE#. Tutti i dati ricevuti ma non letti vengono persi. Per esempio:

CLOSE#7
- chiude il canale della rete e pone a 0 i byte dei dati del flusso.

La figura 3 illustra il protocollo per comunicare i dati attraverso la rete. Anche in questo caso non e' possibile evidenziare lo stato di attesa di una stazione o dell'altra.

DETTAGLI TECNICI DELLA RETE DELLO SPECTRUM

Viene ora spiegato in dettaglio il funzionamento della rete, cio' puo' interessare anche l'utente che usa soltanto i comadi Basic appena spiegati, ma deve essere approfondito solo da chi intende usare o modificare la rete programmando in linguaggio macchina.

La rete usa due fili: uno serve da massa (0 volt) e l'altro porta il segnale che e' attivo alto (+5 volt) e inattivo basso (a massa).

La rete si puo' trovare in due stati, puo' essere cioe' "a riposo" oppure "in uso". Comunque, quando la rete e' in uso, ci sono degli intervalli di inoperosita' che non sono pero' mai lunghi come se fosse a riposo. A

richiedere la rete e' sempre il computer sorgente, mentre il computer destinazione deve semplicemente interpretare tutte le trasmissioni e rispondere quando richiesto.

Lo scambio dei dati da una stazione all'altra avviene secondo un protocollo rigoroso. Viene ora preso in considerazione il caso che lo Spectrum UTENTE debba trasmettere uno o piu' buffer di dati alla stazione AMICO. Per il momento non si esaminano i dettagli della temporizzazione che saranno illustrati alla fine del capitolo.

RICHIESTA DELLA RETE

Questo e' il primo passo di ogni comunicazione, infatti la rete potrebbe essere sia a riposo che in uso da parte di un altro Spectrum. Lo Spectrum UTENTE deve innanzitutto accertarsi che la rete sia in stato di riposo, controllando per un certo periodo di tempo che non avvenga alcuna trasmissione. Il periodo di controllo e' formato dal tempo teoricamente necessario piu' una certa quantita' di tempo extra preso a caso.
Accertatosi che la rete e' libera, UTENTE invia a bassa velocita' un bit di start seguito dal suo numero di stazione a otto bit. Il valore trasmesso e' quello contenuto in NTSTAT e rappresenta il numero di stazione globale e non deve essere per forza lo stesso contenuto in NCSELF. I bit del numero di stazione vengono trasmessi invertiti, bit piu' significativo prima. Ogni bit porta la rete a livello alto se e' 0 e a livello basso se e' 1. Prima di trasmettere il bit successivo viene effettuato un controllo per verificare che la rete sia nello stato in cui dovrebbe essere e, se non lo e', viene ripetuta da capo la procedura di richiesta.
Quando tutti i bit del numero di stazione sono stati trasmessi (e hanno superato il controllo) la rete puo' essere considerata in uso da parte dello Spectrum UTENTE. Il byte del numero di stazione costituisce uno SCOUT; tutte le stazioni in ascolto sulla rete quando ricevono il LEADER dello SCOUT attivano i temporizzatori in base ai quali ispezionano la rete per controllarne lo stato.

La figura 4 mostra le forme d'onda dello SCOUT trasmesso dalla stazione 10.

PRIMA:
(a) GLI OPERATORI SI METTONO D'ACCORDO DI USARE I NUMERI DI STAZIONE 1 & 2;
(b) GLI OPERATORI CONCORDANO CHE LA STAZIONE 1 INVIERA' UN INSIEME DI NUMERI ALLA STAZIONE 2.

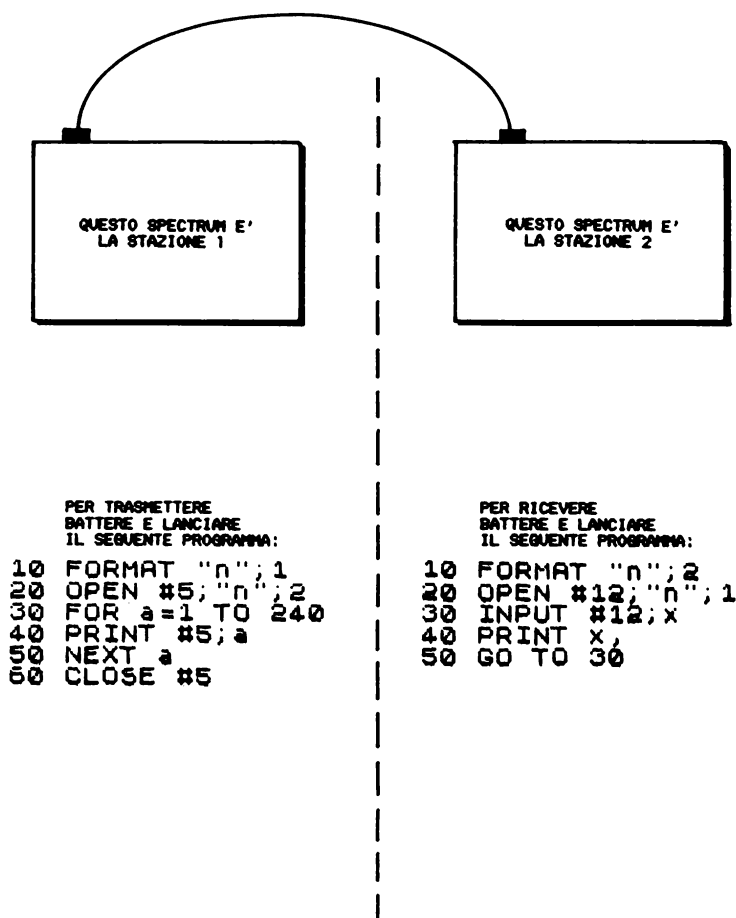



FIG.3. TRASMISSIONE DI DATI ATTRAVERSO LA RETE

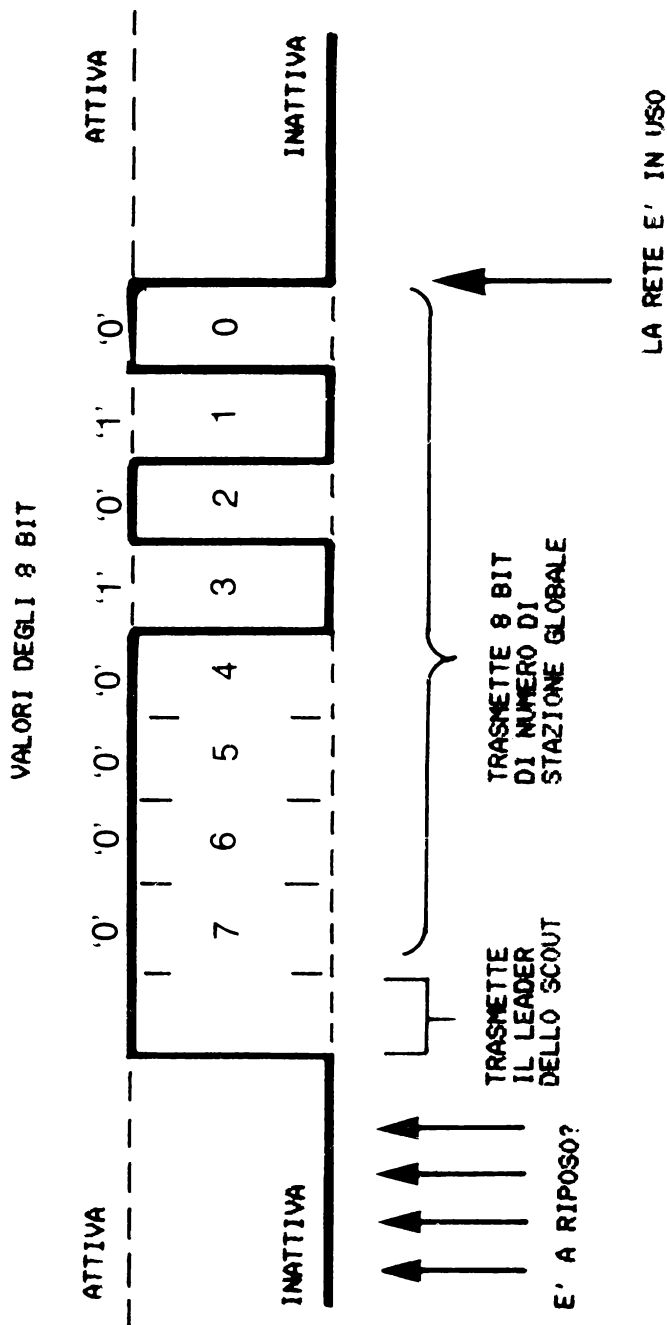



FIG.4. LA STAZIONE 10 TRASMETTE IL SUO SCOUT

INVIO DELL'INTESTAZIONE

Dopo che la rete e' stata dichiarata in uso (con l'invio dello SCOUT), viene trasmessa un'intestazione di 8 byte. I dati dell'intestazione sono prelevati dal canale della rete.

Il canale della rete ha la seguente struttura:

```
BYTE      CONTENUTO
0-1       Indirizzo 0008h
2-3       Indirizzo 0008h
4         "N"              ("N" + 80h se 'ad hoc')
5-6       Indirizzo OUT-N  (indirizzo di ouput)
7-8       Indirizzo IN-N   (indirizzo di input)
9-10      Numero 0114h     (276 decimale: numero di byte
                           del canale della rete)
;   i byte trasmessi come intestazione sono contenuti
;   nelle posizioni 11-18:
11        NCIRIS           (numero della stazione AMICO)
12        NCSELF           (proprio numero di stazione, il
                           valore contenuto in NTSTAT
                           quando viene creato il canale)
13-14     NCNUMB           (numero progressivo del blocco
                            nell'intervallo 0-65535)
15        NCTYPE           (di solito 0 che indica un
                           blocco ordinario di dati;
                           1 indica il blocco di fine file)
16        NCOBL            (lunghezza del buffer di output,
                           nell'intervallo 0-255; e' 0
                           quando il buffer e' usato per
                           ricevere dati)
17        NCDCS            (8 bit di checksum per i dati)
18        NCHCS            (8 bit di checksum dei 7 byte
                           precedenti)
;   le due locazioni successive sono usate per
;   ricevere dati:
19        NCCUR            (posizione dell'ultimo carattere
                           letto dal buffer)
20        NCIBL            (numero di byte che possono
                           essere letti dal buffer
                           dei dati)
;   il seguente buffer dei dati viene usato sia per
;   ricevere che per trasmettere:

21-275    NCB              (buffer dei dati di 255 byte)
```

I byte dell'intestazione descrivono il blocco dei dati che sara' trasmesso subito dopo. Il blocco dei dati e' una copia del buffer al momento della trasmissione.

I byte dell'intestazione vengono inviati chiamando la subroutine OUTPAK contenuta nella ROM fantasma (incorporata nell'interfaccia ZX1). Il registro "E" deve contenere il valore 08h e il registro "HL" l'indirizzo di NCIRIS.

La subroutine OUTPAK invia i dati come segue:

- un LEADER iniziale attivo

poi, per ogni byte:

- un bit di START a livello basso

- un bit a livello alto o basso a seconda che il corrispondente bit del dato originale sia settato o resettato. Il bit meno significativo viene inviato per primo.

- un segnale di stop a livello alto

e, terminati i byte:

- la rete viene posta a livello basso.

La figura 5 evidenzia il procedimento.

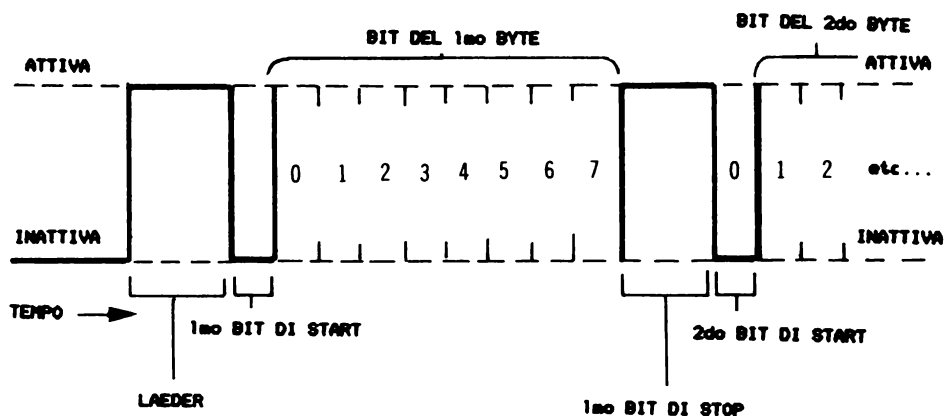



FIG.5. BYTE DI DATI TRASMESSI DALLA SUBROUTINE OUTPACK

Dopo che e' stata trasmessa l'intestazione viene esaminato il valore di NCIRIS e, se e' diverso da 0 si entra in un ciclo di attesa della risposta da parte di AMICO. Uno 0 indicherebbe una trasmissione collettiva a cui non viene mai data risposta.

La risposta e' un unico byte di dati trasmesso da AMICO tramite la stessa sbroutine OUTPAK. Lo Spectrum UTENTE la considera positiva se si tratta di un byte di valore 1 ricevuto da AMICO entro il tempo opportuno.

Per ricevere la risposta viene utilizzata la subroutine INPAK. All'atto della chiamata di questa subroutine il registro "E" deve contenere il valore 01h e la coppia di registri "HL" l'indirizzo 23757 della variabile di sistema fantasma NTRESP.

Se non viene ricevuta una risposta affermativa entro il tempo prestabilito viene ripetuta l'intera operazione di richiesta della rete, viene inviato nuovamente lo SCOUT e l'intestazione, e viene ancora attesa la risposta. In questa fase il bordo dello schermo rimane del colore contenuto nella variabile IOBORD. Quando la subroutine INPAK ritorna un byte = 1 viene trasmesso il blocco dei dati.

TRASMISSIONE DI UN BLOCCO DI DATI

Il buffer dei dati viene trasmesso chiamando la subroutine OUTPAK. Con il registro "E" che contiene la lunghezza attuale del buffer, contenuta nella variabile NCOBL, e la coppia di registri "HL" che contiene l'indirizzo del primo byte del buffer, l'indirizzo NCB.

Dopo che e' stato trasmesso il blocco dei dati, viene esaminata la variabile NCIRIS e, se il blocco dei dati non fa parte di una trasmissione collettiva, viene attesa una risposta. Anche in questo caso se non viene ricevuta la risposta corretta entro il tempo prestabilito viene ripetuta l'intera operazione di trasmissione, compresa la richiesta della rete.

NUMERO PROGRESSIVO DEL BLOCCO

I blocchi di dati sono numerati da 0 a 65535. La creazione di un canale della rete assegna automaticamente al primo blocco il numero 0 e i blocchi seguenti vengono

numerati in ordine crescente. La trasmissione di un insieme di blocchi di dati, quale potrebbe essere quella di un lungo programma Basic, viene divisa in blocco numero 0, blocco numero 1, ..., blocco numero "n", dove "n" e' il numero del blocco di fine file.

Vengono ora esaminate le operazioni necessarie per ricevere uno o piu' blocchi di dati.

IDENTIFICAZIONE DELLO SCOUT

Il primo passo necessario per stabilire una comunicazione da AMICO a UTENTE e' di identificare uno SCOUT (che non deve provenire necessariamente da AMICO). Lo SCOUT viene rivelato quando la rete diventa attiva dopo un periodo di riposo. In questa fase e' necessario che UTENTE esamini ripetutamente la rete per il tempo sufficiente (senza alcun tempo supplementare, vedi prima), per essere sicuro che si trova in stato di riposo, e quindi aspetti una transizione a livello alto. E' AMICO che trasmette il LEADER dello SCOUT, dichiarando in uso la rete. Il numero di stazione, che AMICO manda come SCOUT, non e' richiesto da UTENTE, che quindi aspetta che la rete ritorni inattiva dopo la trasmissione degli otto bit.

Ovviamente qualunque Spectrum in ascolto sulla rete riceve lo SCOUT.

RICEZIONE DELL'INTESTAZIONE

Poco dopo aver inviato lo SCOUT il computer AMICO invia un'intestazione di 8 byte che il computer UTENTE legge con la subroutine INPACK. Questi byte sono caricati nelle variabili di sistema fantasma NTDEST - NTHCS, indirizzi 23758-23765. La figura 6 evidenzia la lettura dei byte dalla rete. La cosa piu' importante da puntualizzare e' che la lettura di ogni bit nel suo "punto di mezzo" viene sincronizzata dall'elettronica dell'interfaccia.

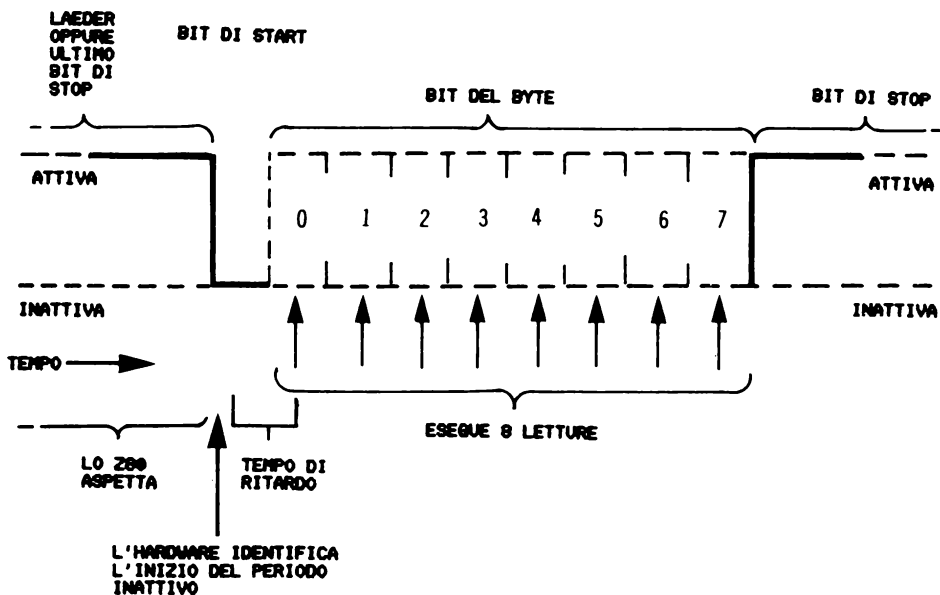



FIG.6. RICEZIONE DI UN BYTE DA PARTE DELLA SUBROUTINE INPACK

ESAME DELL'INTESTAZIONE

Una volta ricevuta, l'intestazione viene esaminata per decidere se accettare o meno il blocco dei dati che seguira'.

In pratica vengono poste una serie di domande e una sola risposta negativa fa si che SCOUT, intestazione e blocco di dati vengano ignorati fino allo SCOUT successivo. Ecco le domande:

i. Sono stati effettivamente letti 8 bit?

ii. La somma dei primi 7 byte da' lo stesso valore contenuto nell'ottavo byte, ovvero e' corretto il checksum dell'intestazione (NTHCS)?

iii. E' il numero di blocco contenuto in NTNUMB quello atteso? (se il numero di blocco e' lo stesso del precedente significa che AMICO sta ritrasmettendo un blocco che UTENTE ha gia' ricevuto ma di cui evidentemente si e' persa la risposta. In questo caso il

blocco viene accettato per non interrompere la trasmissione ma non viene messo in memoria.

iv. Se si sta effettuando una trasmissione collettiva, l'intestazione e' parte di una trasmissione collettiva, cioe' sono NTDEST e NCIRIS entrambi 0? Altrimenti, con una determinata stazione AMICO, l'intestazione specifica che il blocco dei dati e' per UTENTE? cioe' sono NTDEST e NCSELF uguali?

v. La stazione AMICO sta usando il numero di stazione previsto, cioe' sono uguali NTSRCE e NCIRIS?

RICEZIONE DI UN BLOCCO DI DATI

Se non si tratta di una trasmissione collettiva, AMICO inviera' un blocco di dati quando e se ricevera' una risposta da UTENTE prima del tempo prestabilito.

Il computer UTENTE riceve questo blocco di dati caricando nel registro "E" la lunghezza del blocco di dati, cioe' il valore di NTLEN, e nella coppia di registri "HL" l'indirizzo base del buffer dei dati del canale della rete in uso, cioe' l'indirizzo NCB, e chiamando la subroutine INPAK.

Una volta ricevuto, la somma dei byte del nuovo buffer viene confrontata col byte di checksum (contenuto in NTDCS) e, se i due numeri coincidono, viene inviata la risposta. Se il checksum risulta errato, la stazione AMICO riprova a trasmettere i dati e la stazione UTENTE li rilegge.

Se tutto si svolge senza errori il buffer dei dati del canale della rete contiene ora i 255 byte appena trasmessi.

CONTESA TRA SPECTRUM CHE CERCANO DI AGGIUDICARSI LA RETE

E' ovvio che non ci devono essere conflitti se due Spectrum cercano di aggiudicarsi la rete allo stesso istante. L'uso di un tempo extra determinato a caso prima di riconoscere lo stato di riposo e l'invio del numero di stazione globale come SCOUT, aiutano a risolvere eventuali contese. Comunque, se due o piu' Spectrum dovessero trasmettere uno SCOUT contemporaneamente, lo

Spectrum che trasmette il numero di stazione globale piu' basso dovrebbe sempre riuscire ad aggiudicarsi la rete, mentre gli altri dovrebbereo ritirarsi e riprovare in seguito. Gli utenti possono facilitare questo assegnando ad ogni stazione un numero globale diverso.

AMBIGUITA' RICEVENDO TRASMISSIONI COLLETTIVE

E' possibile che se piu' di una stazione ha in corso una trasmissione collettiva allo stesso tempo, la stazione ricevente puo' a volte ricevere un blocco di dati da una ed il successivo dall'altra. Questo e' dovuto al fatto che i blocchi di una trasmissione collettiva sono inviati molto distanziati nel tempo uno dall'altro per essere sicuri che la stazione ricevente abbia abbastanza tempo per elaborare i dati ricevuti; anche se a causa di questo la stazione ricevente spreca molto tempo in ascolto sulla rete.

L'unico modo di evitare il problema e' di prestare attenzione inviando una grossa quantita' di dati sul canale di trasmissione collettiva.

TEMPORIZZAZIONE DELLA FUNZIONE INKEY$

Una stazione in ascolto sulla rete a causa di un comando LOAD, VERIFY, MERGE o INPUT, rimane in attesa per un tempo indefinito fino a quando non sarano ricevuti SCOUT, intestazione e blocco dei dati attesi. Invece con la funzione INKEY$ la stazione aspetta soltanto un certo tempo prefissato e poi ritorna una stringa nulla se non e' stato ricevuto nessun carattere.

PARTICOLARITA' DEL MESSAGGIO DI FINE FILE (END OF FILE) NELLA TRASMISSIONE DEI DATI

Quando vengono aggiunti dei byte di dati al buffer del canale della rete, viene tenuta nota della lunghezza del buffer (ovvero della quantita' di caratteri presenti) nella variabile NCOBL, mentre il valore di NCTYPE rimane sempre 0. Al tentativo di scrivere il 256mo byte il

buffer viene trasmesso come blocco di 255 byte di dati ordinari. Se invece il buffer contiene l'ultimo blocco di dati, la variabile NCTYPE viene posta ad 1 e quindi il buffer diventa il blocco di fine file, che contiene fino a 255 caratteri.

L'operazione di marcare un blocco di dati come blocco di fine file avviene soltanto:

- al termine di un'operazione di SAVE;

- chiudendo un flusso con CLOSE#.

PARTICOLARITA' DEL MESSAGGIO DI FINE FILE (END OF FILE) NELLA RICEZIONE DEI DATI

Quando dei byte di dati vengono letti nel buffer del canale della rete, viene incrementato il valore della variabile NCCUR fino a quando non ci sono piu' byte non ancora utilizzati. Di solito alla successiva richiesta di carattere viene caricato un altro blocco di dati, ma questo avviene soltanto se NTTYPE ha valore 0. Se NTTYPE ha valore 1 si trattava del blocco di fine file e si e' quindi cercato di leggere un carattere oltre l'ultimo; allora viene visualizzato il relativo messaggio di errore ("end of file").

TEMPORIZZAZIONE DELLA RETE

Lo Spectrum funziona ad una frequenza di clock di 3,5 MHz. Nei paragrafi seguenti si fa riferimento ai cicli T del microprocessore, ognuno dei quali dura 1/3.500.000 di secondo.

TRASMISSIONE DI DATI

La rete viene considerata in stato di riposo se e' a livello basso per circa 10.500 cicli T (3 ms.).

Il LEADER dello SCOUT inizia 22 cicli T dopo l'ultimo esame della rete e dura 181 cicli T. Gli 8 bit del numero

di stazione vengono pure trasmessi in 181 cicli T per ognuno. Gli impulsi vengono riletti per il controllo dopo 136 cicli T.

La rete e' a questo punto considerata in uso e il segnale di inizio degli 8 byte dell'intestazione arriva dopo altri 271 cicli T.

I byte dell'intestazione sono trasmessi dalla subroutine di OUTPAK tutti con lo stesso formato:

LEADER - 98 cicli T

per ogni byte:

bit di START - 40 cicli T
8 bit - 40 cicli T ognuno
bit di STOP - 145 cicli T (ma l'ultimo
periodo di STOP e' di 86 cicli T).

Il blocco dei dati segue l'intestazione dopo circa 600 cicli T nel caso di una trasmissione collettiva, ma, se deve essere ricevuta e verificata una risposta, e' ritardato fino a 9000 cicli T.

La durata del blocco dei dati varia tra 544 cicli T (un byte) e circa 128.000 cicli T (255 byte = 37 msec.). La figura 7 illustra queste temporizzazioni.

RICEZIONE DI DATI

Le temporizzazioni per la ricezione dei dati devono essere obbligatorimente le stesse della trasmissione, ma qualche osservazione puo' essere utile.

La rete viene esaminata ripetutamente per circa 7000 cicli T (2 msec.) per verificare che sia in stato di riposo.

Dopo viene esaminata una volta ogni 55 cicli T fino a quando viene trovata attiva. Lo SCOUT viene quindi identificato entro il primo terzo della durata del suo LEADER. Siccome i dati dello SCOUT non interessano, lo Spectrum attende fino a quando non e' stato trasmesso completamente.

Viene effettuata una chiamata alla subroutine INPAK per ricevere i successivi 8 byte di intestazione. Questa subroutine viene chiamata durante il periodo inattivo (271 cicli T) che separa lo SCOUT e l'intestazione.

La rete viene quindi esaminata ogni 35 cicli T per identificare la presenza del LEADER dell'intestazione.

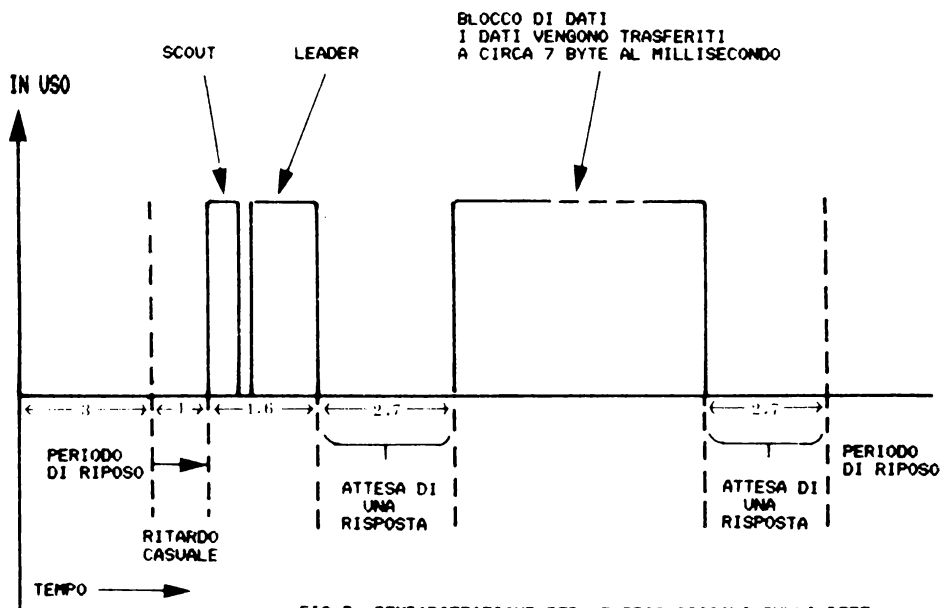



FIG.7. TEMPORIZZAZIONE PER LE TRASMISSIONI SULLA RETE

Quindi l'elettronica di sincronizzazione dell'interfaccia ZX1 identifica il fronte di discesa all'inizio del bit di START. C'e' sempre una certa imprecisione in questa operazione, ma essendo effettuata dall'hardware essa avviene con la maggior precisione possibile.

Il resto del bit di START viene ignorato e quindi vengono letti gli 8 bit del byte alla velocita' di un bit ogni 40 cicli T. Esiste ovviamente una certa imprecisione dato che i clock dei due Spectrum potrebbero avere una frequenza leggermente diversa. Il sistema di sincronizzazione sul periodo di START consente di ovviare ad uno scarto tra le velocita' di clock fino al 5%.

I byte dell'intestazione, una volta ricevuti, vengono esaminati. Se e' in corso una trasmisione collettiva non viene trasmessa nessuna risposta e lo Spectrum si pone in attesa del LEADER del blocco di dati entro 600 cicli T dalla fine del blocco di intestazione. Invece, se lo Spectrum che ha inviato i dati e' in attesa di una risposta, questa deve essere inviata entro approssimativamente 9000 cicli T.

Il blocco dei dati viene ricevuto tramite la subroutine INPAK e, anche in questo caso, lo Spectrum ricevente ha 9000 cicli T di tempo per inviare la sua risposta.

C A P I T O L O   5

I N T E R F A C C I A   R S 2 3 2

INTRODUZIONE

La terza possibilita' offerta dall'interfaccia ZX1 e' il collegamento RS232.
I due usi piu' immediati dell'interfaccia RS232 sono:

- collegare lo Spectrum ad una stampante (diversa dalla ZX Printer);
- collegare lo Spectrum ad un altro computer (magari ad un altro Spectrum, in alternativa alla rete locale).

Ma l'interfaccia puo' essere usata per collegare lo Spectrum a qualunque altra periferica compatibile RS232.

USO DELL'INTERFACCIA RS232

La velocita' con cui vengono trasferiti i dati tra due dispositivi collegati all'interfaccia RS232 viene chiamata "baud rate"; all'accensione la baud rate dell'interfaccia ZX1 viene settata a 9600 baud.
La velocita' teorica alla quale i byte vengono trasferiti e' data dall'espressione:

baud rate/11   byte al sec.

Comunque un byte di dati viene trasmesso soltanto quando la periferica a cui e' destinato e' pronta a riceverlo (stato ready) e in piu' ci devono essere intervalli tra un byte e l'altro; ne consegue che la velocita' reale di trasmissione di un blocco di dati e' in pratica molto inferiore a quella teorica.

SETTARE LA BAUD RATE

La baud rate viene normalmente settata da un comando FORMAT; per esempio:

FORMAT "b";110

e possono essere ottenuti i seguenti valori: 50, 110, 300, 600, 1200, 2400, 4800, 9600, 19200.

Si noti che la baud rate piu' bassa e' 50 e la piu' alta e' 19200 e che se il valore specificato nel comando FORMAT e' fuori da questo intervallo, la baud rate viene settata al valore limite relativo. Per esempio:

FORMAT "b";0 - setta la baud rate a 50.

In piu', se il valore specificato nel comando FORMAT e' intermedio tra i valori permessi viene usata la baud rate immediatamente inferiore. Per esempio:

FORMAT "b";1500 - setta la baud rate a 1200

L'unico effetto del comando FORMAT e' di scrivere il valore opportuno nei due byte della variabile di sistema BAUD; e' anche possibile settare la baud rate con comandi POKE o con le istruzioni in linguaggio macchina "LD". In questi casi i valore di BAUD e' dato dall'espressione:

(3500000/(26*baud rate)) -2

In quest'ultimo modo e' possibile settare la baud rate sia ad un valore standard, sia ad un valore non standard.

TRASMETTERE PROGRAMMI, MATRICI E DATI BINARI

I programmi Basic, le matrici e i dati binari possono essere trasmessi attraverso l'interfaccia RS232 con il comando SAVE. Per esempio:

SAVE*"b"
- viene trasmessa una copia del programma in memoria e delle sue variabili.

SAVE*"b" SCREEN$
- viene trasmessa una copia del display file e degli attributi.

LOAD, MERGE E VERIFY CON PROGRAMMI, MATRICI E DATI BINARI

I programmi Basic, le matrici o i dati binari possono essere ricevuti tramite l'interfaccia RS232. Per esempio:

LOAD*"b"
- carica un programma Basic e le sue variabili.
MERGE*"b"
- fonde col programma in memoria un programma Basic e le sue variabili.
VERIFY*"b" DATA 32000,100
- controlla il blocco di dati binari ricevuti con il contenuto attuale della parte di RAM specificata.

In tutti i casi, usando SAVE, LOAD, MERGE o VERIFY occorre usare "b" o "B" come indicatore di periferica; i byte trasmessi possono avere un qualunque valore tra 0 e 255 (decimale).
Si noti anche che i dati vengono trasmessi sull'interfaccia RS232 solo se la periferica cui sono destinati segnala che e' pronta a riceverli. Come con la rete il colore del bordo dello schermo durante i trasferimenti di dati ed i periodi di attesa e' determinato dal valore della variabile di sistema IOBORD che puo' essere modificata a piacere.

TRASMISSIONE DATI

Come con la rete locale, la trasmissione di dati richiede l'apertura di un flusso, associato all'interfaccia RS232, prima di poter usare il comando PRINT. Per esempio:

OPEN#4;"b"
- apre il flusso numero 4 per l'uso con l'interfaccia RS232.

Un flusso cosi' aperto puo' essere usato sia per ricevere che per trasmettere byte di dati. Questo e' possibile dato che per l'interfaccia RS232 non esiste alcun buffer e ogni byte viene gestito individualmente.

L'apertura di un flusso per l'interfaccia RS232 produce l'aggiunta di 11 byte di dati all'area delle informazioni di canale. Questi byte sono:

BYTE	CONTENUTO
1-2	Indirizzo 0008h
3-4	Indirizzo 0008h
5	"T" ("T" + 80 se 'ad hoc')
6-7	Indirizzo OUT-T2 (indirizzo di output)
8-9	Indirizzo IN-T2 (indirizzo di input)
10-11	Numero 11 (undici byte nel canale)

Le routine della ROM fantasma, IN-T2 e OUT-T2, gestiscono la ricezione e la trasmissione dei dati.

Per comunicare dati, dopo che e' stato aperto un flusso per l'interfaccia RS232, si possono usare i seguenti comandi:

Per Trasmettere:

PRINT #4;A
- trasmette il valore contenuto nella variabile "A" seguito dal carattere "ENTER";
PRINT #4; "stringa di caratteri"
- trasmette la stringa di caratteri seguita dal carattere "ENTER".

Per ricevere:

INPUT #4;A$
- tutti i caratteri ricevuti prima del carattere "ENTER" vengono assegnati ad A$.
LET A$ = INKEY$#4
- il primo carattere ricevuto viene assegnato ad A$. Si noti che CODE A$ sara' 0 se in quel momento non c'e' alcun carattere disponibile.

Il comando CLOSE,per esempio CLOSE#4, produce la stampa di un "line feed" (carattere avanzamento linea, 0Ah), prima della chiusura del canale di 11 byte.

Si noti che se il "line feed" in piu' e' origine di problemi e' meglio non chiudere i flussi che si usano con l'interfaccia RS232 ma usare CLEAR# o provocare una situazione di errore che chiude il canale.

IL SISTEMA "T"

Gli indicatori di periferica "t" e "T" permettono un uso diverso dell'interfaccia RS232, consentendo di trasmettere i caratteri ASCII e i "token" Sinclair in modo testo. Questo modo e' utile perche' permette la trasmissione di un listato con il comando LIST#n dove "n" e' un flusso associato con l'interfaccia RS232.

Si puo' produrre il listato su una stampante esterna con, per esempio:

- OPEN#4;"T"
- LIST#4
- CLOSE#4 - il "line feed" e' probabilmente necessario.

In questo "modo testo" i codici dei caratteri sono modificati come segue:

- i codici di controllo (00h-1Fh) sono ignorati, salvo il ritorno di carrello (0Dh) che e' modificato e diventa "ritorno di carrello" + "line feed" (0Dh e 0Ah)
- i codici dei caratteri grafici vengono cambiati in "?" (3Fh)
- i codici dei "token" Sinclair (parole chiave) vengono espansi e ricondotti ai caratteri ASCII.

USO ESTESO DEL SISTEMA "T"

Ci sono tre modi per collegare lo Spectrum ad una stampante compatibile RS232.

i. Usando un solo flusso di tipo "B", che permette di trasferire byte di qualunque valore (00h-FFh). Questo metodo lascia all'utente il compito di gestire i codici di controllo, i caratteri grafici e i token, ma non pone limiti su quello che puo' essere ottenuto.

ii. Usando sia un flusso "T" che un flusso "B". In questo modo l'utente puo' gestire i caratteri stampabili sul flusso "T" e inviare i caratteri di controllo richiesti sul flusso "B". Per esempio, per dare il comando ESCAPE alla stampante si potrebbe usare:

DA FOTOGRAFARE; Pag. 76

```
  10 OPEN #4;"t": OPEN #5;"b": R
EM flussi "T" & "B"
  20 PRINT #5;CHR$ 27;CODE "k";:
 REM = 'escape k'
  30 PRINT #4;"stampa questo"
```

iii. Il terzo metodo richiede l'uso del flusso "T" esteso. Come gia' menzionato, il flusso "T" ignora tutti i byte nell'intervallo 00h-1Fh (salvo 0Dh, "ENTER"). Questo fa si che l'utente non sia in grado di sfruttare i controlli di posizione (virgola, "AT" e "TAB") nei comandi PRINT associati al flusso "T" RS232, anzi, vengono creati dei caratteri spuri se AT e TAB sono seguiti da parametri di valore elevato.

Il seguente programma Basic mostra come il flusso "T" puo' essere modificato per poter trasmettere questi controlli di posizione se l'utente intende usarli in un comando PRINT.

```
  10 OPEN #2;"t": CLEAR 65000: G
O SUB 9800
  20 REM ... la vostra stampa va
 inserita qui ...
  30 STOP
9800 REM #2, AT TAB subroutine
9810 REM usa Z, Z1, Z2, Z3
9820 REM setta i TAB e il numero
 di colonne
9830 POKE 65000,0: POKE 65001,40
9840 REM esamina il flusso #2
9850 LET z1=PEEK 23578+256*PEEK
23579
9860 IF NOT z1 THEN POKE 23610,2
3: STOP
9870 REM esamina il canale "T"
9880 LET z2=z1+PEEK 23631+256*PE
EK 23632+3
9890 IF PEEK z2<>CODE "t" THEN P
OKE 23610,23: STOP
9900 REM modifica gli indirizzi
di output
9910 POKE z2+1,234
9920 POKE z2+2,253
9930 REM scrive il linguaggio ma
cchina
```

```
9940 FOR z=2 TO 169
9950 READ z3
9960 POKE 65000+z,z3
9970 NEXT z
9980 RETURN
9990 DATA 254,165,48,21,254,32,5
6,20,245,58,233,253,79,58,232,25
3,60,185,56,1,175,50,232,253,241
,195,60,12
9991 DATA 254,13,32,8,175,50,232
,253,62,13,24,241
9992 DATA 254,6,32,31,237,75,232
,253,30,0,12,28,121,184,40,8,214
,10,40,4,48,250,24,242,213,62,32
,205,234,253,209,29,200,24,245
9993 DATA 254,22,40,12,254,23,40
,8,254,16,216,17,96,254,24,3,17,
88,254,50,14,92,42,81,92,213,17,
5,0,25,209,115,35,114,201,17,96,
254,50,15,92,24,235,17,234,253,2
05,75,254,87,58,14,92,254,22,40,
8,254,23,63,192,58,15,92,87,58,2
33,253,186,40,2,48,6,71,122,144,
87,24,242,58,232,253,186,200,213
,62,32,205,234,253,209,24,242
```

NOTE AL PROGRAMMA

Linea 10 - viene utilizzato il flusso 2; viene caricata una routine in linguaggio macchina alle locazioni 65000-65156 e un caricatore Basic alle linee 9800-9993.

Linea 9830 - il numero di colonne puo' essere modificato come si desidera - qui e' settato a 40 colonne. L'attuale posizione di stampa puo' essere rilevata leggendo la locazione 65000.

Linea 9990 - la posizione di stampa viene incrementata ogni volta che viene stampato un carattere, fino a quando e' stata riempita una linea, allora viene resettata a 0.

Linea 9991 - anche un ritorno di carrello (ENTER) resetta la posizione di stampa.

Linea 9992 - gestisce la virgola della PRINT. Le posizioni a cui la virgola fa saltare in questo programma sono 0, 10, 20 e 30; il valore dell'incremento di 10 (che si trova in questa linea) puo' essere alterato a piacere. La stampa di una virgola produce sempre l'avanzamento della posizione di stampa di almeno una colonna.

Linea 9993 - questa linea gestisce "AT" e "TAB". I parametri di questi comandi vengono presi modulo il numero di colonne (contenuto nella locazione 65001). Tutti gli indicatori di colore sono ignorati.

Nota: questo programma produce l'effetto di inviare la stampa, diretta al video, verso l'interfaccia RS232.

DETTAGLI TECNICI DELL'INTERFACCIA RS232

Il collegamento RS232 attraverso l'interfaccia ZX1, permette sia di trasmettere che di ricevere dati seriali verso o da una periferica compatibile RS232 oppure un altro Spectrum.

La trasmissione dei dati attraverso l'interfaccia RS232 avviene byte per byte (e non e' bufferizzata come la rete), ma ogni byte viene trasmesso soltanto se la periferica ricevente segnala che e' pronta a riceverlo.

Il collegamento RS232 e' formato da 6 fili. Due vengono usati per ricevere, due per trasmettere, il quinto e' un filo di massa e il sesto porta una tensione di 9v., ma di solito non viene impiegato.

PROTOCOLLO DI TRASMISSIONE

Il filo usato per la trasmissione dei dati si chiama RXdata (received data line = linea dei dati ricevuti) e il livello del segnale su questo e' determinato dal dispositivo trasmittente. Il secondo filo coinvolto nell'operazione di trasmissione viene chiamato DTR (data terminal ready = terminale dei dati pronto) e viene settato dal dispositivo ricevente.

L'operazione di trasmissione di un byte di dati puo' essere suddivisa nelle seguenti operazioni:

i. Attesa fino a quando la linea DTR diventa alta.

ii. Trasmissione del byte di dati.

I due passi vengono ripetuti per ogni byte di dati.
Di solito il dispositivo ricevente porta a livello basso il segnale DTR quando ha ricevuto il byte, evitando cosi' che il dispositivo trasmittente possa trasmettere un altro byte prima che possa essere ricevuto (vedi dopo).
La forma d'onda del segnale RXdata e' disegnata in figura 1.
Si noti che per ogni byte di dati vengono trasmessi 11 bit. Cioe':

- un bit di start
- 8 bit di dati
- due bit di stop (possono essere considerati come un bit di lunghezza doppia)

mentre non viene trasmesso il bit di parita'.

IL PROTOCOLLO DI RICEZIONE

I dati vengono ricevuti attraverso il filo chiamato TXdata (transmitted data line = linea dei dati trasmessi) il cui livello viene determinato dal dispositivo trasmittente. Il secondo filo coinvolto nell'operazione viene chiamato CTS (clear to send line = libero per la trasmissione) e viene settato dal dispositivo ricevente.

Ne consegue che se due Spectrum sono collegati insieme attraverso l'interfaccia RS232, i segnali RXdata, DTR, TXdata e CTS di uno Spectrum devono essere connessi rispettivamente ai segnali TXdata, CTS, RXdata e DTR dell'altro Spectrum.

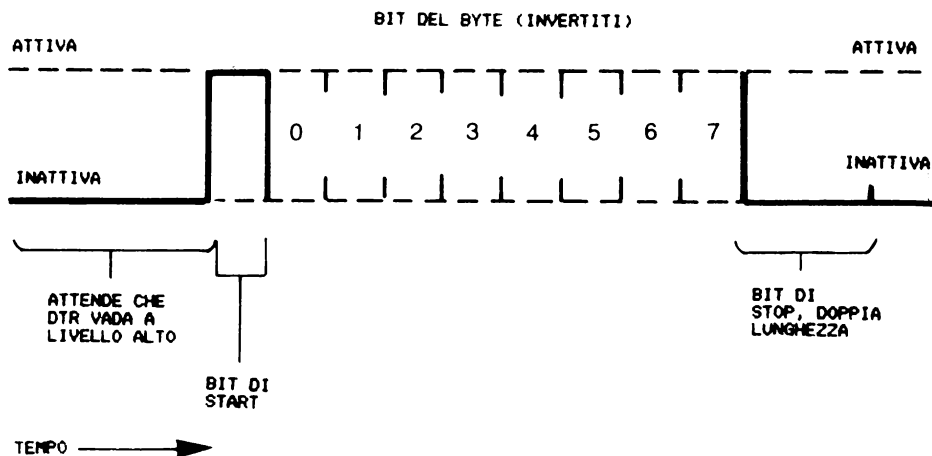
L'operazione di ricezione di un byte di dati puo' essere suddivisa nelle seguenti operazioni:

NOTA: LUNGHEZZA DI UN BIT = 47 + 26*BAUD STATI "T".

FIG.1. UN BYTE DI DATI TRASMESSO ATTRAVERSO L'INTERFACCIA RS232

i. Innanzitutto esamina la variabile fantasma di sistema SER-FL (serial flat). Se il valore nel byte meno significativo e' diverso da 0, allora il byte richiesto e' contenuto nel byte piu' significativo (e' gia' stato ricevuto; vedi dopo); lo preleva e azzera il byte meno significativo di SER-FL prima di abbandonare la subroutine. Diversamente, se il valore e' 0, riceve uno o due byte di dati.

ii. Porta la linea CTS a livello alto.

iii. Aspetta fino a quando il segnale sulla linea TXdata va a livello alto, cio' indica l'inizio del bit di start.

iv. Legge gli 8 bit del byte.

v. Memorizza tutto il byte.

vi. Porta la linea CTS a livello basso.
Nota: per quanto la linea CTS sia ora a livello basso e' possibile che il dispositivo trasmittente mandi un altro byte di dati, questo comunque non accade se si tratta di uno Spectrum.

vii. Ripete i passi 3 e 4 se viene inviato un altro byte. Questo secondo byte va sistemato nella seconda locazione della variabile di sistema SER-FL e la prima locazione settata a 1. La regola e' che se non viene ricevuto un altro byte entro un certo periodo di tempo, viene assunto che non sara' trasmesso e SER-FL rimane 0.

viii. Il byte memorizzato al passo 5 viene richiamato, e la routine abbandonata.

TEMPORIZZAZIONI DELL'INTERFACCIA RS232

La durata di ogni bit trasmesso e' data dall'espressione:

lunghezza del bit = 47 + 26*BAUD

espressa in intervalli T, dove ogni T vale 1/3500000mo di secondo.

Il valore di BAUD si ottiene con la seguente espressione:

BAUD = (3500000/(baud rate*26))-2

Con baud rate = 19200 ogni bit dura teoricamente:

= 3500000/19200
= 182,3 T

mentre la velocita' reale e'

BAUD = INT (3500000/(19200*26))-2
= 7

e la lunghezza di un bit e':

= 47 + 26*BAUD
= 177 T

Da tutto questo si deduce che alla velocita' di 19200 lo scarto tra velocita' teorica e velocita' reale e' inferiore al 3%, e a velocita' piu' basse lo scarto e' piu' ridotto.

COLLEGAMENTO CON COMPUTER DIVERSI DALLO SPECTRUM

L'interfaccia RS232 permette di collegare lo Spectrum a qualunque altro computer dotato a sua volta di tale interfaccia. Questo significa che altri calcolatori possono usare il Microdrive, la rete locale e la ZX Printer usando uno Spectrum come Controller. Si potrebbe realizzare un'interfaccia dedicata ad un certo calcolatore, per esempio per il Microdrive, ma usare uno Spectrum e' senz'altro piu' economico.

I paragrafi seguenti descrivono a grandi linee come collegare un computer Acorn BBC ad un sistema Spectrum.

Il computer ha un connettore a 5 poli RS423 che e' direttamente compatibile con il connettore RS232 dello Spectrum. Per stabilire una comunicazione bidirezionale occorre collegare i segnali TXdata, RXdata, DTR, CTS e GND dello Spectrum rispettivamente alle linee "data out", "data in", RTS, CTS e GND del computer BBC. La baud rate, e' all'accensione 9600 su entrambe le macchine.

USO DELLA ZX PRINTER

La cosa piu' semplice e' usare la ZX Printer sotto il controllo del computer BBC.

Nello Spectrum occorre soltanto dare il comando:

```
MOVE"b" TO #3 e ENTER
```

dopo aver naturalmente collegato la stampante e il cavo RS232.

Nel microcomputer BBC occorre scrivere per la stampa diretta:

```
*FX5,2          - seleziona la periferica seriale
CTRL B          - abilita la stampante
...
...             - tutti i messaggi vanno alla
                  ZX printer
CTRL C          - disabilita la stampante
```

Mentre in un programma, per esempio:

```
10 *FX5,2       - seleziona il seriale
20 VDU2         - = CTRL B
30 *FX21,2      - elimina tutti i caratteri dal
                  buffer RS423
...
...             - la stampa (si usi *FX3,7 se non
                  e' richiesto anche l'output
                  sullo schermo)
100 VDU3        - = CTRL C
```

USO DEL MICRODRIVE

Gli esempi seguenti illustrano come il computer BBC puo' usare il Microdrive per memorizzare un file di dati.

Nota: e' possibile gestire programmi Basic, trattandoli come un insieme di byte di dati ma l'autore non e' in grado di eseguire SAVE e LOAD attraverso l'interfaccia RS423 del computer BBC, anche se questa operazione puo' essere probabilmente svolta con una routine in linguaggio macchina.

CREAZIONE DI UN FILE SUL MICRODRIVE

Nel computer BBC occorre scrivere:

```
 10 DIMA$(2)                ;crea qualche dato
 20 LETA$(0)="aaaa"
 30 LETA$(1)="bbbb"
 40 LETA$(2)="cccc"
 50 *FX5,2                  ;seleziona la
                             stampante seriale
 60 VDU2                    ;prepara a trasmettere
 70 PRINT 3                 ;tre, il numero dei
                             dati
                             che formano il file
 80 PRINTA$(0)              ;invia i dati
 90 PRINTA$(1)
100 PRINTA$(2)
110 VDU3                    ;conclude la
                             trasmissione
```

Sullo Spectrum occorre scrivere:

```
10 OPEN #4;"b"              ;legge dal flusso #4
20 OPEN #5;"M";1;"BBC1"     ;memorizza nel flusso
                             #5
30 INPUT #4;A               ;carica il numero
                             dei dati
40 FOR N=1 TO A
50 INPUT #4;A$              ;carica ogni dato
60 PRINT #5;A$              ;lo trasmette al
                             buffer del microdrive
70 NEXT N                   ;
80 CLOSE #4                 ;chiude il flusso
                             di input
90 CLOSE #5                 ;crea il file sul
                             microdrive
                            ;e chiude il canale
```

E' naturalmente possibile fare si' che il nome del file venga comunicato dal computer BBC e che il primo elemento del file sia il numero di dati contenuti.

LETTURA DI UN FILE DAL MICRODRIVE

L'operazione inversa potrebbe essere (assumendo che il file BBC1 abbia tre elementi):

Nello Spectrum:

```
10 OPEN#5;"m";1;"bbc1"     ;leggi dal flusso 5
20 OPEN#4;"b"              ;trasmetti sul
                            flusso 4
30 FOR N=1 TO 3
40 INPUT#5;A$              ;carica ogni dato
50 PRINT#4;A$              ;trasmetti ogni dato
60 NEXT N
70 CLOSE#4
80 CLOSE#5
```

e, nel computer BBC, si potrebbe usare il seguente programma per ricevere i dati dal collegamento RS423:

```
10 *FX2,1                      ;seleziona l'input
                                RS423
20 *FX3,6                      ;seleziona "no output"
30 *FX21,2                     ;inizializza il buffer
                                RS423
40 DIMA$(2)                    ;ci sono tre elementi
                                da leggere
50 FORN=0TO2
60 INPUTR$                     ;usa R$ per leggere
                                ogni dato
70 LETA$(N)=R$                 ;trasferisce i dati
80 NEXTN
90 *FX3,4                      ;seleziona l'output
                                sullo schermo
100*FX2,0                      ;seleziona l'input
                                sulla tastiera
...
110 PRINT A$(0),A$(1),A$(2)
```

Questi esempi vogliono significare che non e' affatto complicato usare il Microdrive per memorizzare file con calcolatori collegati allo Spectrum attraverso l'interfaccia RS232.

Infine se l'utente vuole ricevere i dati byte per byte invece di usare INPUT puo' usare il seguente programma sul microcomputer BBC:

```
10 FX2,1  - seleziona l'ingresso RS423 *FX2,1
20 A%=145
30 X%=1
40 R=USR(&FFF4) - legge un singolo byte
50 R=(R AND &00FF0000)/65536
60 IF R<>0 THEN PRINT CHR$(R);
70 GOTO 10 - ripete per il byte successivo
```

NOTE

Linea 50 - isola il valore ASCII dal valore ritornato dalla funzione USR.

Linea 60 - il valore contenuto in R puo'essere manipolato come piu' e' opportuno.

L'ultimo programma Basic complementa il programma gia' definito nella sezione "Uso esteso del sistema T" e puo' essere utilizzato per ricevere i dati trasmessi dallo Spectrum sull'interfaccia RS232.

C A P I T O L O 6

U S O D E L L I N G U A G G I O M A C C H I N A

NOTA DELL'AUTORE

Chi non ha familiarita' con il linguaggio macchina dello Z80 non si deve preoccupare di leggere questo capitolo in dettaglio. Un buona conoscenza del linguaggio macchina e' tuttavia essenziale per poter essere in grado di scrivere le proprie routine di controllo. Si consiglia, a quanti volessero apprenderlo, di acquistare uno dei tanti libri dedicati alla programmazione in linguaggio macchina dello Z80 per principianti, meglio se con qualche riferimento allo Spectrum. L'autore ha pubblicato sull'argomento: "Understanding Your Spectrum".

E' molto utile avere a disposizione un editor o un assembler per il linguaggio macchina; l'autore preferisce il programma "SPDE" della Campbell Systems si tratta di un editor-disassembler che puo' essere trasferito su Microdrive, e' molto pratico da usare, anche se e' lento in confronto ai veri e propri assembler.

INTRODUZIONE

Questo capitolo e' diviso in due parti:

a. Come usare i codici di chiamata delle routine della ROM fantasma.

b. Come aggiungere nuove routine.

Queste due operazioni non hanno assolutamente niente in comune.

A. USO DEI CODICI DI CHIAMATA

Nello Spectrum gli errori vengono gestiti dalla routine RESTART 0008h. Gli errori sono numerati, nello Spectrum non esteso, nell'intervallo FFh - 1Ah. La gestione degli errori avviene eseguendo l'istruzione RST 0008h seguita da un DEFB (definizione byte) contenente il valore appropriato.
Per esempio:

```
                      ORG    7D00H      ;32000 DEC.
7D00 CF        EXAM   RST    0008H      ;RESTART DI GESTIONE ERRORI
7D01 0B               DEFB   0BH        ;= "NONSENSE IN BASIC"
                      END
```

Che in Basic diventa:

POKE 32000,207:POKE 32001,11:RANDOMIZE USR 32000

Il valore di DEFB fuori dall'intervallo prestabilito produce una situazione di errore ma il messaggio e' privo di significato, infatti ci sono messaggi significativi solo per i valori consentiti.

In uno Spectrum dotato di interfaccia ZX1 quando viene eseguito un prelevamento di istruzione all'indirizzo 0008h viene inserita la ROM fantasma. Nel programma della ROM fantasma che comincia a questo indirizzo, il valore di DEFB viene analizzato come segue:

- se DEFB e' nell'intervallo 00h - 1Ah si ritorna alla ROM principale.
- se DEFB e' nell'intervallo 1Bh - 32h si tratta dei codici di accesso alle subroutine.
- se DEFB e' nell'intervallo 33h - FEh si ha il messaggio di errore "hook code error" (codice accesso errato).
- se DEFB e' 00h si ha il messaggio di errore "program finished" (programma finito).

I valori dell'intervallo 1Bh - 32h consentono all'utente di chiamare diverse subroutine nella ROM fantasma. In pratica l'uso dell'istruzione RST 0008h e del codice di chiamata DEFB e' equivalente a chiamare la routine XXXX della ROM fantasma.

Ogni volta che viene effettuata questa chiamata, viene

innanzitutto inserita la ROM fantasma, poi eseguita la subroutine ed infine la ROM viene disinserita. L'utente non puo' mai lasciare la ROM fantasma attiva.

Vengono ora discussi individualmente i codici 1Bh e 32h, ma prima si noti che non viene conservato intatto il contenuto di alcun registro, quindi si faccia attenzione a salvare i registri significativi prima di usare un codice di chiamata. Con le subroutine piu' complicate occorre salvare anche H'L' che contiene l'indirizzo di ritorno al Basic.

Durante tutte le operazioni di input/output gli interrupt mascherabili sono disabilitati ed anche alcune routine fantasma disattivano questi interrupt, mentre altre richiedono che siano gia' disabilitati prima della loro chiamata.

Il codice di accesso 32h e' riservato alla Sinclair Research Limited.

Il codice di accesso 31h inserisce le nuove variabili di sistema fantasma in memoria e non si deve esitare a ricorrervi quando e' necessario (vedi dopo).

CODICE DI ACCESSO 1BH - INPUT DA CONSOLE

Questa subroutine aspetta che venga premuto un tasto sulla tastiera dello Spectrum. Quando questo accade, ne viene ritornato il codice nel registro A. Gli interrupt mascherabili devono essere attivi. I tasti che non producono caratteri vengono ignorati.

Puo' essere usato come segue:

```
                        ORG     7D00H     ;32000 DEC.
7D00 3E02       CON#IN  LD      A,02H     ;PER STAMPA NELLA PARTE ALTA
7D02 CD0116             CALL    1601H     ;CHIAMA L'APERTURA DI CANALE
7D05 06FF               LD      B,0FFH    ;INIZIALIZZA UN CONTATORE
7D07 C5         EACH    PUSH    BC        ;SALVA IL CONTATORE
7D08 CF                 RST     0008H     ;CHIAMA "INPUT DA CONSOLE"
7D09 1B                 DEFB    1BH       ;PER CARICARE OGNI CARATTERE
7D0A C1                 POP     BC        ;RIPRISTINA IL CONTATORE
7D0B D7                 RST     0010H     ;STAMPA IL CARATTERE
7D0C 10F9               DJNZ    EACH      ;DECREMENTA E RIPETE SE <> 0
7D0E C9                 RET               ;RITORNA
                        END
```

In questo esempio la tastiera viene letta 255 volte e i caratteri letti stampati sullo schermo.

CODICE DI ACCESSO 1CH - OUTPUT SU CONSOLE

Questa subroutine esegue la stessa operazione di - LD A,02h - CALL SELECT - RST 0010h, che stampa il carattere del registro A verso il flusso 2, cioe' la parte alta dello schermo televisivo. Si noti che lo scrolling viene soppresso. L'esempio precedente puo' essere modificato come segue:

```
                     ORG    7D00H     ;32000 DEC.
7D00 06FF   CON#OUT  LD     B,0FFH    ;INIZIALIZZA UN CONTATORE
7D02 C5     EACH     PUSH   BC        ;SALVA IL CONTATORE
7D03 CF              RST    0008H     ;CHIAMA "INPUT DA CONSOLE"
7D04 1B              DEFB   1BH       ;PER CARICARE OGNI CARATTERE
7D05 CF              RST    0008H     ;CHIAMA "OUTPUT SU CONSOLE"
7D06 1C              DEFB   1CH       ;PER STAMPARE OGNI CARATTERE
7D07 C1              POP    BC        ;RIPRISTINA IL CONTATORE
7D08 10F8            DJNZ   EACH      ;DECREMENTA E RIPETE SE <> 0
7D0A C9              RET              ;RITORNA
                     END
```

CODICE DI ACCESSO 1DH - INPUT DALL'INTERFACCIA RS232

Questa subroutine riceve dei dati attraverso l'interfaccia RS232. La velocita' di trasferimento dei dati e' controllata dal valore di BAUD e il colore del bordo dallo schermo dal valore di IOBORD.

E' opportuno resettare SER-FL prima di cominciare a ricevere dati, dato che l'ultimo carattere precedentemente ricevuto potrebbe ancora esservi contenuto.

Il seguente esempio mostra come si possono ricevere

caratteri usando questo codice di accesso.
Si ricordi che SER-FL non esiste prima dell'inserimento delle variabili fantasma di sistema. Si usi il codice 31h se richiesto (vedi dopo).

```
                    ORG     7D00H          ;32000 DEC.
7D00 AF     RS#IN   XOR     A              ;AZZERA IL REGISTRO
7D01 32C75C         LD      (SER#FL),A     ;RESETTA SER-FL
7D04 06FF           LD      B,0FFH         ;INIZIALIZZA UN CONTATORE
7D06 C5     EACH    PUSH    BC             ;SALVA IL CONTATORE
7D07 CF             RST     0008H          ;CHIAMA "INPUT DA RS232"
7D08 1D             DEFB    1DH            ;PER CARICARE OGNI CARATTERE
7D09 CF             RST     0008H          ;CHIAMA "OUTPUT SU CONSOLE"
7D0A 1C             DEFB    1CH            ;PER STAMPARE OGNI CARATTERE
7D0B C1             POP     BC             ;RIPRISTINA IL CONTATORE
7D0C 10F8           DJNZ    EACH           ;DECREMENTA E RIPETE SE <> 0
7D0E C9             RET                    ;RITORNA
                    END
```

CODICE DI ACCESSO 1EH - OUTPUT SU RS232

Questa e' la corrispondente subroutine di output. Anche qui la velocita' di trasferimento dati viene determinata da BAUD e il colore del bordo dello schermo da IOBORD. Come al solito non vengono inviati se DTR non e' a livello alto.
Il seguente esempio trasmette i dati ricevuti dal programma precedente.

```
                    ORG     7D00H          ;32000 DEC.
7D00 06FF   RS#OUT  LD      B,0FFH         ;INIZIALIZZA UN CONTATORE
7D02 C5     EACH    PUSH    BC             ;SALVA IL CONTATORE
7D03 3E41           LD      A,41H          ;CARATTERE "A"
7D05 CF             RST     0008H          ;CHIAMA "OUTPUT SU RS232"
7D06 1E             DEFB    1EH            ;PER TRASMETTERE 255 "A"
7D07 C1             POP     BC             ;RIPRISTINA IL CONTATORE
7D08 10F8           DJNZ    EACH           ;DECREMENTA E RIPETE SE <> 0
7D0A C9             RET                    ;RITORNA
                    END
```

Questo esempio trasmette un insieme di 255 byte di valore 41h.

CODICE DI ACCESSO 1FH - OUTPUT SU ZX PRINTER

Questa subroutine e' identica a quella dell'output su console ma usa il flusso #3 invece del flusso #2 e quindi di solito indirizza i dati in uscita verso la ZX Printer.

CODICE DI ACCESSO 20H - TEST DELLA TASTIERA

Questa subroutine usa le seguenti istruzioni per controllare la pressione di un tasto.

```
AF          XOR   A          ;AZZERA IL REGISTRO A
DBFE        IN    A,(0FEH)   ;LEGGE 8 LINEE DELLA TASTIERA
E61F        AND   1FH        ;CONSIDERA I 5 BIT MENO SIGNIF.
D61F        SUB   1FH        ;NESSUN TASTO = 0
                             ;QUALUNQUE TASTO = NEGATIVO
C6FF        ADD   A,0FFH     ;NESSUN TASTO = CARRY RES.
                             ;QUALUNQUE TASTO = CARRY SET.
            END
```

L'uso di - RST 0008h & DEFB 20h - invece di queste cinque istruzioni non e' di alcuna convenienza pratica.

CODICE DI ACCESSO 21H - SELEZIONE DEL DRIVE

Questo e' il primo dei 13 codici di accesso relativi ai Microdrive.

Questa subroutine avvia il motore del Microdrive specificato, dove il numero del Microdrive puo' essere nell'intervallo 1-8. La selezione del Microdrive 0 produce l'arresto di tutti i motori accesi. Se viene avviato un motore la subroutine ritorna gli interrupt mascherabili disabilitati.

Se l'operazione non puo' essere eseguita la subroutine produce un messaggio di errore "Microdrive not present".

Per esempio il Microdrive 1 puo' essere avviato da:

```
                    ORG    7D00H      ;32000 DEC.
7D00 3E01   SEL#DR  LD     A,01H      ;PER MICRODRIVE 1
7D02 CF             RST    0008H      ;CHIAMA SELEZIONE DEL DRIVE
7D03 21             DEFB   21H
7D04 FB             EI                ;ABILITA GLI INTERRUPT
7D05 C9             RET               ;RITORNA
                    END
```

Il motore puo' poi essere fermato dal Basic producendo un qualsiasi errore, per esempio con CAT e ENTER.

CODICE DI ACCESSO 22H - APERTURA DI UN FILE

Questo codice permette di creare un canale "ad hoc" dal linguaggio macchina. L'indirizzo base del canale viene ritornato nel registro IX.
Per usare questo codice occorre:

- assicurarsi che le variabili di sistema fantasma siano state create (se necessario si usi il codice di accesso 31h - vedi dopo);
- salvare il contenuto di H'L';
- scrivere il numero del drive in D-STR1;
- scrivere il nome del file nelle locazioni appropriate;
- scrivere il descrittore del file in N-STR1; in ordine 1. lunghezza (LSB), 2. lunghezza (MSB), 3. indirizzo di partenza (LSB), 4. indirizzo di partenza (MSB);
- chiamare la routine "Open file";
- ripristinare H'L' prima di ritornare al Basic.

Questo codice di accesso puo' essere usato soltanto per creare un canale per il file di dati e funziona in

modo analogo al comando OPEN.

- se il nome del file e' nuovo il file viene aperto per la scrittura, altrimenti viene aperto per la lettura e viene caricato il primo record del file.

E, una volta creato il file, puo' essere gestito cosi':

- trasformare il canale del microdrive in canale corrente, per es. con: PUSH IX, POP HL, LD(CURCHL), HL;

e poi o:

- scrivere un file usando RST 0010h; con i caratteri nel registro A;

oppure:

- leggere il file usando CALL 15E6 (INCH o INPUT-AD).

Si noti che la scrittura del 513mo, 1025mo, ... carattere provoca la creazione di un nuovo record, in modo analogo vengono caricati i record richiesti durante la lettura.
Un esempio di apertura di un file viene dato nel seguente esempio di chiusura di un file.

CODICE DI ACCESSO 23H - CHIUSURA DI UN FILE

L'azione di questa subroutine e' molto simile a quella del comando CLOSE associato ad un file di dati sul Microdrive. Se il file, il cui indirizzo base e' contenuto nel registro IX, e' aperto per la scrittura, fa si' che ogni dato ancora non inviato vada a formare il record di fine file e che venga chiuso il canale, ma se il file e' aperto per la lettura il canale viene chiuso e ogni dato nella sua area viene perso.
L'esempio seguente mostra la creazione di un canale per un file di nome "a" sul Microdrive 1. Questo file viene scritto oppure letto prima di essere chiuso.

```
                ORG     7D00H           ;32000 DEC.
        NAME    EQU     7CFFH           ;CONTERRA' "A"
CF      COMMON  RST     0008H           ;CREA LE VARIABILI DI SISTEMA
31              DEFB    31H             ;SE NON SONO GIA' STATE CREATE
D9              EXX
E5              PUSH    HL              ;SALVA IL VALORE DI H'L'
D9              EXX
3E01            LD      A,01H           ;SELEZIONA IL DRIVE 1
32D65C          LD      (D#STR1),A
3E61            LD      A,61H           ;IL NOME DEL FILE E' "A"
32FF7C          LD      (NAME),A
210100          LD      HL,0001H        ;"A" E' UN CARATTERE SINGOLO
22DA5C          LD      (N#STR1),HL
21FF7C          LD      HL,NAME         ;INDIRIZZO DI PARTENZA = NOME
22DC5C          LD      (N#STR1+02H),HL
        :
CF      FILE    RST     0008H           ;CHIAMA "APERTURA DI UN FILE"
22              DEFB    22H
DDE5            PUSH    IX              ;RENDE IL CANALE DI MICRODRIVE
E1              POP     HL              ;IL CANALE CORRENTE
22515C          LD      (CURCHL),HL
     ;VIENE LETTO O CREATO UN FILE DI 3000 BYTE
01B80B          LD      BC,0BB8H        ;3000 DEC.
C5      EACH    PUSH    BC              ;SALVA IL CONTATORE
     ;PER LA SCRITTURA USARE ESCLUSIVAMENTE QUESTI CODICI
3E41            LD      A,41H           ;IL FILE VIENE RIEMPITO
D7              RST     0010H           ;CON 3000 "A"
     ;PER LA LETTURA USARE ESCLUSIVAMENTE QUESTI CODICI
CDE615          CALL    INCH            ;CARICA UN CARATTERE
CF              RST     0008H           ;CHIAMA "OUTPUT SU CONSOLE"
1C              DEFB    1CH
DDE5            PUSH    IX              ;RENDE ANCORA IL CANALE DI
E1              POP     HL              ;MICRODRIVE CORRENTE
22515C          LD      (CURCHL),HL
     ;IN ENTRAMBI I CASI USARE I SEGUENTI CODICI
C1              POP     BC              ;CARICA IL CONTATORE
0B              DEC     BC              ;DECREMENTA IL CONTATORE
78              LD      A,B             ;E' ZERO?
```

```
B1              OR     C
20__            JR     NZ,EACH       ;RIPETE SE <> 0
CF              RST    0008H         ;CHIAMA "CHIUSURA DI UN FILE"
23              DEFB   23H
3E02            LD     A,02H         ;RISELEZIONA IL VIDEO
CD0116          CALL   SELECT
D9              EXX                  ;CARICA IL VALORE DI H'L'
E1              POP    HL            ;PRECEDENTEMENTE SALVATO
D9              EXX
C9              RET                  ;RITORNA
                END
```

Il lettore deve usare questo programma innanzitutto per creare un nuovo file sul drive 1 e poi rileggere i dati scritti. Il salto deve essere "20F6" per la scrittura e "20EE" per la lettura.

CODICE DI ACCESSO 24H - CANCELLAZIONE DI UN FILE

Questa subroutine puo' essere usata per cancellare un file con un certo nome dalla cartuccia del Microdrive. I parametri del file devono essere preparati in D-STR1 e N-STR1. Con questa subroutine possono essere cancellati sia i file di dati che i file programmi. Per esempio il file 'a' sul drive 1 puo' essere cancellato come segue: (ma fatene una copia per dopo usando il comando MOVE "m";1;"a" TO "m";1;"b")

```
                       ORG    7D00H         ;32000 DEC.
                NAME   EQU    7CFFH         ;CONTERRA' "A"
         ;INSERIRE LE STESSE 13 LINEE DI PROGRAMMA DI "CHIUSURA FILE"
7D1B CF         DELETE RST    0008H         ;CHIAMA "CANCELLAZIONE FILE"
7D1C 24                DEFB   24H
7D1D D9                EXX
7D1E E1                POP    HL            ;RIPRISTINA H'L'
7D1F D9                EXX
7D20 C9                RET                  ;RITORNA
                       END
```

CODICE DI ACCESSO 25H - LETTURA SEQUENZIALE

Questo codice permette all'utente di caricare il record di un certo file di dati nell'area di lavoro del suo canale di Microdrive. All'atto della chiamata il registro IX deve contenere l'indirizzo base del canale di Microdrive e la variabile CHREC il numero progressivo del record correntemente in memoria, questo numero viene incrementato automaticamente.

Il seguente esempio carica in ordine i 6 record del file di testo "a":

```
                          ORG     7D00H        ;32000 DEC.
7CFF            NAME.     EQU     7CFFH        ;CONTERRA' "A"
        ;INSERIRE LE STESSE 13 LINEE DI PROGRAMMA DI "CHIUSURA FILE"
7D1B CF         READSQ    RST     0008H        ;CHIAMA "APERTURA FILE"
7D1C 22                   DEFB    22H
7D1D CD407D     EACH      CALL    REC#NO       ;STAMPA IL NUMERO DI RECORD
7D20 CF                   RST     0008H        ;CHIAMA "LETTURA SEQUENZIALE"
7D21 25                   DEFB    25H
7D22 18F9                 JR      EACH         ;PER OGNI RECORD
                ;
                          ORG     7D40H        ;32064 DEC.
        ;SUBROUTINE DI STAMPA DEL NUMERO DI RECORD
7D40 DD7E44     REC#NO    LD      A,(IX+44H)   ;LEGGE IL NUMERO CORRENTE
7D43 C630                 ADD     A,30H        ;TRASFORMA IN ASCII
7D45 CF                   RST     0008H        ;CHIAMA "OUTPUT SU CONSOLLE"
7D46 1C                   DEFB    1CH
7D47 DDE5                 PUSH    IX           ;RENDE CORRENTE
7D49 E1                   POP     HL           ;IL CANALE DI MICRODRIVE
7D4A 22515C               LD      (CURCHL),HL
7D4D C9                   RET                  ;RITORNA
                          END
```

Questo esempio produce solo la stampa di "012345" ma questo significa che i record sono stati caricati anche se non letti. Si noti come il programma si arresti alla fine del file.

CODICE DI ACCESSO 26H - SCRITTURA DI UN RECORD

Questo codice permette all'utente di creare un record. Come al solito il registro IX deve contenere l'indirizzo base del canale di Microdrive che contiene i dati del record. Il record viene creato nel primo settore libero sul nastro. Per esempio:

```
            ORG     7D00H           ;32000 DEC.
      NAME  EQU     7CFFH           ;CONTERRA' "A" - O ALTRO
   ;SCRIVERE LE LINEE DI "CHIUSURA FILE" PER "SCRITTURA"
   ;MODIFICANDO "LD BC,0BB8H" IN "LD BC,0200H"
   ; E "CF 23" (HEX), "CHIUSURA FILE" IN "CF 26", "SCRITTURA FILE"
   ;CAMBIARE ANCHE IL NOME DEL FILE SE SI USA UNA
   ;CARTUCCIA CHE CONTIENE GIA' IL FILE "A"
            END
```

E' opportuno usare la subroutine di chiusura del file quando si scrive il record di fine file.

CODICE DI ACCESSO 27H - LETTURA CASUALE

Questa subroutine e' simile alla lettura sequenziale ma viene caricato il record il cui numero e' contenuto in CHREQ.

CODICE DI ACCESSO 28H - LETTURA DI UN SETTORE

Questa subroutine carica il record del settore specificato CHREQ. Il flag di carry viene resettato se il checksum dei dati e' corretto e settato altrimenti l'utente deve selezionare il Microdrive prima di usare questo codice o i successivi due.

CODICE DI ACCESSO 29H - LETTURA DEL SUCCESSIVO

Questa subroutine carica il record del primo settore che passa davanti alla testina di lettura del Microdrive. Il numero di settore viene posto in HDNUMB. Anche qui il flag di carry viene ritornato resettato soltanto se il checksum dei dati e' corretto. Si noti che non c'e' abbastanza tempo tra i settori per caricare e trasferire un intero record prima di raggiungere l'intestazione del blocco. Quindi i record di un dato file non sono mai sui settori contigui.

CODICE DI ACCESSO 2AH - SCRITTURA DI UN SETTORE

Questa subroutine e' complementare alla lettura di settore (28h). I dati correntemente contenuti nel buffer dei dati vengono scritti nel settore il cui numero e' contenuto in CHREQ.

CODICE DI ACCESSO 2BH - CREAZIONE DI UN BUFFER

Questa subroutine doveva in teoria creare un canale di Microdrive (e una mappa) per i file specificati da D-STR1 e N-STR1. Purtroppo a causa di un errore di programmazione questo codice di accesso ha lo stesso effetto del codice di apertura file (22h). Questo errore verra' corretto nelle future ROM.

I seguenti esempi mostrano comunque come puo' essere creato un canale di Microdrive e una mappa.

```
7C80                    ORG     7C80H                 ;31872 DEC.
                  ;BYTE DEL CANALE
7C80 08                 DEFB    08H
7C81 00                 DEFB    00H
7C82 08                 DEFB    08H
7C83 00                 DEFB    00H
7C84 CD                 DEFB    0CDH                  ;CD E' "M" + 80H
7C85 D8                 DEFB    0D8H                  ;(BUFFER "AD HOC")
7C86 11                 DEFB    11H
7C87 22                 DEFB    22H
7C88 11                 DEFB    11H
7C89 53                 DEFB    53H
7C8A 02                 DEFB    02H
7C8B 00                 DEFB    00H
7C8C 00                 DEFB    00H
7C8D 00                 DEFB    00H
7C8E 61                 DEFB    61H        ;NOME DI FILE "A"
7C8F 20                 DEFB    20H
7C90 20                 DEFB    20H
7C91 20                 DEFB    20H
7C92 20                 DEFB    20H
7C93 20                 DEFB    20H
7C94 20                 DEFB    20H
7C95 20                 DEFB    20H
7C96 20                 DEFB    20H
7C97 20                 DEFB    20H
7C98 FF                 DEFB    0FFH       ;FF E' APERTO PER LA SCRITTURA
7C99 01                 DEFB    01H        ;01 E' PER IL MICRODRIVE 1
7C9A 00                 DEFB    00H
7C9B 00                 DEFB    00H
7C9C 00                 DEFB    00H
7C9D 00                 DEFB    00H
7C9E 00                 DEFB    00H
7C9F 00                 DEFB    00H
7CA0 00                 DEFB    00H
7CA1 00                 DEFB    00H
7CA2 00                 DEFB    00H
```

```
7CA3 00        DEFB   00H
7CA4 00        DEFB   00H
7CA5 00        DEFB   00H
7CA6 FF        DEFB   0FFH
7CA7 FF        DEFB   0FFH   ;FF FF E' LA FINE DELL'INTESTAZIONE
7CA8 00        DEFB   00H    ;PREAMBOLO
7CA9 00        DEFB   00H
7CAA 00        DEFB   00H
7CAB 00        DEFB   00H
7CAC 00        DEFB   00H
7CAD 00        DEFB   00H

        ;I DUE BYTE SOTTOLINEATI SONO CHMAP
        ;E DEVONO ESSERE SETTATI A PARTE

7CAE 00        DEFB   00H
7CAF 00        DEFB   00H
7CB0 00        DEFB   00H
7CB1 00        DEFB   00H
7CB2 00        DEFB   00H
7CB3 00        DEFB   00H
7CB4 00        DEFB   00H
7CB5 00        DEFB   00H
7CB6 00        DEFB   00H
7CB7 00        DEFB   00H
7CB8 00        DEFB   00H
7CB9 00        DEFB   00H
7CBA 00        DEFB   00H
7CBB 00        DEFB   00H
7CBC 00        DEFB   00H
7CBD 00        DEFB   00H
7CBE 00        DEFB   00H
7CBF 00        DEFB   00H
7CC0 00        DEFB   00H
7CC1 FF        DEFB   0FFH
```

```
7CC2 FF                 DEFB    0FFH           ;FF FF E' LA FINE DEL BLOCCO
                                               ;DEI DATI DEL PREAMBOLO
                        ORG     7D00H          ;32000 DEC.
7D00 2A4F5C    START    LD      HL,(CHANS)     ;CREAZIONE DELLA MAPPA
7D03 E5                 PUSH    HL             ;SALVA L'INDIRIZZO BASE
7D04 2B                 DEC     HL
7D05 012000             LD      BC,0020H       ;32 LOCAZIONI
7D08 CD5516             CALL    MAKEROOM       ;CREA LO SPAZIO PER LA MAPPA
7D0B E1                 POP     HL
7D0C E5                 PUSH    HL             ;COPIA L'INDIRIZZO BASE
7D0D AF                 XOR     A              ;TUTTI I BIT DEVONO ESSERE 0
7D0E 0620               LD      B,20H
7D10 77        EACH#ONE LD      (HL),A
7D11 23                 INC     HL
7D12 10FC               DJNZ    EACH#ONE
     ;L'INDIRIZZO BASE DELLA MAPPA E' SULLO STACK,
     ;ORA SI PUO' CREARE IL CANALE
7D14 2A535C             LD      HL,(PROG)      ;PRIMA PROG.
7D17 2B                 DEC     HL
7D18 E5                 PUSH    HL
7D19 E5                 PUSH    HL             ;PRENDE LE DUE COPIE;
7D1A DDE1               POP     IX             ;UNA PER IX
7D1C 015302             LD      BC,0253H       ;LUNGHEZZA DEL CANALE
7D1F CD5516             CALL    MAKEROOM       ;CREA L'AREA DEL CANALE
7D22 21807C             LD      HL,7C80H       ;ORA SCRIVE NEL CANALE 67
7D25 D1                 POP     DE             ;DEI BYTE
7D26 014300             LD      BC,0043H       ;ELENCATI PRIMA
7D29 EDB0               LDIR
     ;ORA RIEMPIE CHMAP
7D2B E1                 POP     HL             ;CARICA L'INDIRIZZO
7D2C DD751A             LD      (IX+CHMAP),L
7D2F DD741B             LD      (IX+CHMAP+01H),H
7D32 C9                 RET                    ;RITORNA
                        END
```

CODICE DI ACCESSO 2CH - CANCELLAZIONE DI UN BUFFER

Questa subroutine e' piu' semplice; chiude, cioe' cancella, il canale e la mappa il cui indirizzo base e' contenuto nel registro IX. Conseguentemente tutte le aree di memoria dinamiche sono spostate in giu' di 627 byte.

CODICE DI ACCESSO 2DH - APERTURA DI UN CANALE DELLA RETE

Questo e' il primo dei quattro codici di accesso relativi alla rete locale. La prima subroutine crea un canale della rete di tipo "B". La stazione destinazione deve essere in D-STR1 e il numero di stazione UTENTE in NTSTAT.

Come al solito l'indirizzo base del nuovo canale viene ritornato nel registro IX. Questo canale puo' essere usato per trasmettere o ricevere byte di dati con RTS 0010h o CALL 15E6, ma prima il canale deve essere reso corrente, vedi dopo.

CODICE DI ACCESSO 2EH - CHIUSURA DI UN CANALE DELLA RETE

Questa subroutine puo' essere utilizzata per chiudere un canale della rete il cui indirizzo base e' contenuto nel registro IX. Se il canale e' stato usato per la trasmissione qualunque dato non ancora inviato viene marcato come dato di fine file e trasmesso, ma se il canale e' usato per la ricezione, i dati non ancora letti vengono cancellati.

L'esempio seguente mostra come trasmettere dati dalla stazione 1 alla stazione 2 in linguaggio macchina.

```
                    ORG     7D00H        ;32000 DEC.
7D00 21D65C   STRT#OUT LD      HL,D#STR1
7D03 3602              LD      (HL),02H     ;DESTINAZIONE 2
7D05 CF                RST     0008H        ;"APERTURA CANALE DELLA RETE"
7D06 2D                DEFB    2DH
7D07 DDE5              PUSH    IX           ;RENDE CORRENTE IL
7D09 E1                POP     HL           ;NUOVO CANALE
7D0A 22515C            LD      (CURCHL),HL
         ;INVIA QUALCHE BYTE DI DATI - PER ES.
7D0D 013412            LD      BC,1234H
7D10 C5       EACH#OUT PUSH    BC
7D11 79                LD      A,C
7D12 D7                RST     PRINTA1
7D13 C1                POP     BC
7D14 0B                DEC     BC
7D15 79                LD      A,C
7D16 B0                OR      B
7D17 20F7              JR      NZ,EACH#OUT
         ;ADESSO SEGNALA LA FINE DEL FINE
7D19 CF                RST     0008H        ;"CHIUSURA CANALE DELLA RETE"
7D1A 2E                DEFB    2EH
7D1B C9                RET                  ;RITORNA
                       END
```

I dati possono essere ricevuti come segue (la stazione 1 riceve dalla stazione 2):

```
                     ORG     7D00H      ;32000 DEC.
7D00 21D65C  STARTIN LD      HL,D#STR1  ;AMICO E' ANCORA 2
7D03 3602            LD      (HL),02H
7D05 CF              RST     0008H      ;"APERTURA CANALE DELLA RETE"
7D06 2D              DEFB    2DH
      ;I BYTE VENGONO RICEVUTI IN SEQUENZA
7D07 DDE5    EACH#IN PUSH    IX         ;RENDE IL CANALE CORRENTE
7D09 E1              POP     HL
7D0A 22515C          LD      (CURCHL),HL
7D0D DDE5            PUSH    IX         ;IX VIENE ALTERATO DA INCH
                                        ;QUANDO SI USA LA RETE
7D0F CDE615  TRY     CALL    INCH       ;CERCA DI CARICARE UN BYTE
7D12 30FB            JR      NC,TRY     ;RIPETE SE LA STAZIONE 2 NON
                                        ;E' IN ATTESA DA 1
7D14 FE20            CP      ' '        ;"OUTPUT SU CONSOLE" NON VUOLE
7D16 38F7            JR      C,TRY      ;CODICI DI CONTROLLO
7D18 DDE1            POP     IX         ;RIPRISTINA IX DOPO INCH
7D1A CF              RST     0008H      ;"OUTPUT SU CONSOLE"
7D1B 1C              DEFB    1CH
7D1C 18E9            JR      EACH#IN    ;NON FINISCE MAI
                     END
```

Il canale della rete puo' essere chiuso utilizzando infine la chiusura del canale della rete (2Eh).

CODICE DI ACCESSO 2FH - LETTURA DEL PACCHETTO

Questa subroutine permette all'utente di caricare un particolare SCOUT, un'intestazione e un blocco di dati. Le variabili del canale della rete NCIRIS, NCSELF e NCNUMB devono essere settate ai valori appropriati. Questa subroutine ritorna il carry settato se passa il tempo consentito senza che venga trasmesso alcun dato o se il checksum risulta errato. Come al solito non c'e' limite al tempo di attesa in caso di trasmissione collettiva.

CODICE DI ACCESSO 30H - TRASMISSIONE DI UN PACCHETTO

Questa subroutine permette all'utente di inviare uno SCOUT, un'intestazione e un blocco di dati. All'atto della chiamata il registro A deve contenere 0 per un blocco di dati ordinari e 1 per un blocco di fine file. Sia questo codice che il precedente producono l'incremento di NCNUMB se vengono eseguiti senza errori.

CODICE DI ACCESSO 31H - CREAZIONE DELLE VARIABILI

Questo codice di accesso puo' essere usato per assicurarsi che le variabili di sistema della ROM fantasma vengano inserite in memoria. Il seguente programma Basic evidenzia come possa essere usato.

```
NEW & ENTER
  10 PRINT "prima"
  20 FOR a=23734 TO 23791
  30 PRINT PEEK a;" ";
  40 NEXT a
  50 POKE 32000,207: REM 'RST 00
08h'
  60 POKE 32001,49: REM crea le
variabili di sistema
  70 POKE 32002,201: REM 'RET'
  80 RANDOMIZE USR 32000
  90 PRINT '
 100 PRINT "dopo"
 110 FOR a=23734 TO 23791
 120 PRINT PEEK a;" ";
 130 NEXT a
RUN & ENTER
```

Si ricordi che le variabili di sistema vengono inserite una sola volta e non possono piu' essere rimosse. Se necessario memorizzate questo programma su cassetta, non su Microdrive, e poi scrivete NEW prima di ricaricarlo.

B. COME AGGIUNGERE NUOVI COMANDI

Attraverso il linguaggio macchina e' possibile aggiungere nuove istruzioni al Basic dello Spectrum, ma solo se si usa uno Spectrum dotato di interfaccia ZX1. Questo e' possibile perche' l'utente puo' cambiare gli indirizzi della variabile di sistema VECTOR e quindi intercettare l'interprete Basic. Le regole che determinano l'inserzione di nuovi comandi sono piuttosto complicate, ma una volta comprese si possono avere a disposizione comandi fatti su misura.

L'INTERPRETE BASIC NELLA ROM PRINCIPALE

Innanzitutto e' utile sapere come vengono gestiti i comandi Basic dalle routine in linguaggio macchina contenute nella ROM residente dello Spectrum. Il procedimento e' suddiviso in tre livelli:

a. L'utente scrive un tentativo di linea Basic nel buffer di edit. Questo viene visualizzato con i token espansi nella parte bassa dello schermo televisivo. La linea Basic e' fatta da una serie di comandi Basic. Per semplicita' nel resto della discussione si considereranno linee di un solo comando.

b. La pressione del tasto ENTER causa l'ispezione della correttezza della sintassi nella linea appena inserita. Durante questa ispezione l'interprete Basic identifica il comando (per esempio PRINT in una linea 10 PRINT 4+2) e quindi chiama la routine di controllo per quel comando che si limita per il momento a controllare la correttezza della sintassi.

Se e' corretta il comando viene accettato e viene indirizzato nella posizione giusta nell'area del programma (se si tratta di una linea di un solo comando che comincia con il numero di linea), e si ritorna all'editor per ricevere la linea successiva.

Diversamente viene effettuato un salto alla routine di gestione degli errori, utilizzando

RST 0008h seguito da un byte definito che contiene il codice dell'errore. Questo fa ritornare all'editor con il tentativo di linea nel suo buffer contenente anche l'indicatore di errore di sintassi (?).

c. Il terzo passo riguarda l'esecuzione del comando. Al momento dell'esecuzione il comando Basic viene identificato una seconda volta e quindi viene eseguita la relativa routine di controllo.

Se il comando viene eseguito senza errori viene preso in considerazione il comando successivo.

Se si verifica qualche errore di esecuzione viene chiamata la routine di gestione degli errori, ma questa volta viene stampato un messaggio di errore nella parte bassa dello schermo.

A questo punto risulta chiaro che, perche' un comando venga accettato ed eseguito, la sua routine di controllo deve contenere obbligatoriamente una parte di analisi di sintassi e una parte esecutiva.

L'INTERPRETE BASIC NELLA ROM FANTASMA

I programmi in linguaggio macchina della ROM fantasma sono un'estensione di quelli della ROM principale e vengono chiamati ogni volta che viene attivata la routine di gestione degli errori.

Le routine della ROM fantasma permettono di riconsiderare gli errori segnalati dalla ROM principale. E' cosi' possibile usare anche le routine di controllo della sintassi e di esecuzione per i comandi relativi a Microdrive, rete di lavoro locale e interfaccia RS232.

Nella ROM fantasma ci sono routine che gestiscono comandi che cominciano con:CAT, FORMAT, MOVE, ERASE, OPEN, SAVE, LOAD, VERIFY, MERGE, CLS E CLEAR; (CLOSE viene gestito separatamente dato che ha un suo indirizzo di inserzione della ROM fantasma).

L'analisi semplificata del funzionamento delle routine della ROM fantasma puo' essere:

- In caso di errore identificare il comando della linea.

- Se il comando non e' nella lista fornita ritornare alla ROM principale attraverso l'indirizzo contenuto nella variabile VECTOR.

- Altrimenti saltare alla relativa routine di controllo, abbandonandola poi con l'indirizzo ST-END nella fase di correzione della sintassi, con l'indirizzo END1 nella fase di esecuzione, con restart ROMERR in caso di errore il cui messaggio e' contenuto nella ROM principale oppure con restart SH-ERR in caso di errore il cui messaggio e' contenuto nella ROM fantasma. In tutti i casi viene riabilitata la ROM principale.

LA SINTASSI DEI NUOVI COMANDI

Quando l'utente vuole aggiungere un nuovo comando al Basic dello Spectrum, e' necessario definire un comando che produca un errore di sintassi sia con la ROM principale che con la ROM fantasma. Si noti che un nuovo comando non puo' cominciare con nessuno degli 11 comandi per i quali esistono gia' delle routine nella ROM fantasma (elencati sopra).

Ci sono moltissimi comandi "jolly" che possono essere definiti, ma l'autore suggerisce che il lettore provi a creare diversi comandi elementari prima di cercare di implementare comandi complessi.
Ecco un esempio di possibili comandi semplici:

Usando LINE	- di solito non ha significato, ma puo' diventare un nuovo comando.
Per esempio: 10 LINE	- puo' disegnare una certa particolare linea.
Usando DRAW	- con un nuovo insieme di parametri.
Per esempio: 10 DRAW n	- potrebbe sottolineare n caratteri.
Usando CIRCLE	- con un nuovo set di parametri.
Per esempio: 10 CIRCLE n	- potrebbe disegnare un cerchio di diametro n.

Usando BORDER — con un nuovo set di parametri.

Per es.: 10 BORDER %b,i,p — il separatore % evita l'intervento della ROM principale. Questo comando puo' cambiare contemporaneamente i colori di BORDER, INK e PAPER.

Questi sono solo suggerimenti; si ricordi che e' possibile usare qualunque frase che non passi i controlli di sintassi (e non venga quindi eseguita).

USO DELLA VARIABILE DI SISTEMA FANTASMA VECTOR

La prima osservazione da fare e' che questa variabile dopo l'inserzione contiene il valore 01FCh (496 decimale). Questo e' l'indirizzo fantasma noto come ERR-6.

- Questo puo' essere evidenziato scrivendo:
CAT e ENTER - (o qualunque altra cosa che ne assicuri l'inserzione);
PRINT PEEK 23735+256%PEEK 23736

Le linee della ROM fantasma che usano VECTOR sono:

```
ERR-V   LD HL, (VECTOR)    ;carica l'indirizzo
                           ;vettorizzato

        JP (HL)            ;e' normalmente ERR-6

ERR-6   ...                ;gestisce gli errori
```

La variabile di sistema fantasma VECTOR normalmente contiene l'indirizzo ERR-6 (01F0h) che e' quindi vettorizzato.

Il primo passo per aggiungere i nuovi comandi e' di cambiare i contenuti di VECTOR in modo che vengano effettuati altri controlli prima di produrre un messaggio di errore.

Il seguente programma Basic mostra come modificare gli indirizzi in VECTOR in modo tale che la routine ESTENSIONE diventi operativa.

```
CAT & ENTER                 - assicura l'inserzione
  10 POKE 23735,0           - il vettore contiene 7D00h
  20 POKE 23736,125         - (32000 dec.)
  30 POKE 32000,195         - questa e' la routine
  40 POKE 32001,240         - "ESTENSIONE":
  50 POKE 32002,1           - e' solo JP ERR-6
RUN & ENTER                 ;e lo Spectrum funziona.
```

USO DEL PUNTATORE CH ADD

CH ADD (indirizzo carattere) e' la variabile di sistema principale contenuta nelle locazioni 5C5Dh e 5C5Eh (23645/6 dec.) ed e' usata per indirizzare i caratteri che formano un comando. All'atto della chiamata della routine ESTENSIONE la variabile di sistema CH ADD punta al comando del messaggio di errore.

Le due subroutine della ROM principale RESTART 0018h (GET CHAR) e RESTART 0020h (NEXT CHAR) ritornano all'utente nel registro A rispettivamente il carattere indirizzato da CH ADD e il carattere successivo. Entrambe le routine ignorano gli spazi e i caratteri di controllo.

GET CHAR e NEXT CHAR sono routine molto utili e forniscono un modo conveniente di gestire CH ADD.

LA SUBROUTINE CALBAS

Questa subroutine e' contenuta a partire dall'indirizzo 0010h (16 dec) della ROM fantasma e puo' essere chiamata con l'istruzione RTS 0010h. Questa subroutine permette all'utente di chiamare una subroutine della ROM principale quando e' inserita la ROM fantasma.

L'istruzione RST 0010h deve essere seguita dall'indirizzo della subroutine della ROM principale. Per esempio per chiamare GET CHAR quando e' abilitata la ROM fantasma:

```
D7        RST    CALBRAS
1800      DEFW   GET$CHAR
          END
```

Si noti che tutti i registri vengono ritornati con il loro contenuto originario dalla subroutine nella ROM principale.

MODULI DI SINTASSI E DI ESECUZIONE

La routine di comando che deve essere aggiunta per gestire un nuovo comando deve essere composta da due parti.

IL MODULO DI SINTASSI

Per verificare la sintassi di un comando e' necessario identificare:

- il comando;
- tutti i necessari separatori;
- le espressioni numeriche;
- le espressioni stringa;
- le variabili;

e infine:

- la fine del comando

che deve essere ":" oppure "ritorno di carrello". Le subroutine che possono essere usate per aiutare queste operazioni vengono esaminate negli esempi che seguono.

IL MODULO DI ESECUZIONE

In questo modulo viene eseguito il comando vero e proprio, a seconda di come e' stato implementato.

Vengono ora discussi alcuni semplici comandi aggiuntivi.

ESEMPI PRATICI DI COMANDI AGGIUNTIVI

L I N E

Questo nuovo comando disegna una linea che racchiude in un quadrato lo schermo. Equivalente a:
PLOT 0,0:DRAW 255,0;DRAW 0,175:DRAW -255,0:DRAW 0,-175

MODULO DI SINTASSI

Per questo comando occorre:

- caricare il codice del comando - usando (GET CHAR);
- comparare il codice con quello di LINE (CAh/202 dec.): Se il codice non e' lo stesso saltare a ERR 6, altrimenti procedere con il modulo di sintassi;
- incrementare CH ADD cosicche' punti dopo il comando: si usa NEXT CHAR;
- chiamare la routine ST END per verificare la fine del comando, confermando l'esame positivo della sintassi.

In linguaggio macchina tutto questo e':

```
                        ORG    7D00H        ;32000 DEC.
7D00 D7        EXTEND   RST    CALBAS
7D01 1800               DEFW   GET#CHAR     ;CARICA IL COMANDO
7D03 FECA               CP     LINE         ;E' "LINE"?
7D05 CA207D             JP     Z,LINE#SYN   ;SE SI, SALTA
7D08 C3F001             JP     ERR#6        ;ABILITA GESTIONE ERRORI
               ;
7D20                    ORG    7D20H        ;MODULO DI SINTASSI
                                            ;DEL COMANDO LINE
7D20 D7        LINE#SYN RST CALBAS
7D21 2000               DEFW   NXT#CHAR     ;AVANZA CH-ADD
7D23 CDB705             CALL   ST#END       ;CONFERMA LA FINE DEL COM. &
                                            ;ESCE CONTROLLATA LA SINT.
          ;MODULO DI ESECUZIONE PROVVISORIO
7D26 C3C105    LINE#RUN JP     END1         ;ESCE - NESSUN EFFETTO
                        END
```

Si consiglia al lettore di inserire questi 20 byte di codici, cambiare il valore in VECTOR e provare il comando LINE prima di procedere con il modulo di esecuzione che segue.

MODULO DI ESECUZIONE

L'esecuzione di LINE produce una linea che circonda lo schermo televisivo. La routine per questa funzione e':

```
                        ORG      7D26H          ;32038 DEC.
7D26 210000  LINE#RUN   LD       HL,0000H       ;POSIZIONE DI PLOT SETTATA
7D29 227D5C             LD       (COORDS),HL    ;A (0,0)
7D2C 01FF00             LD       BC,00FFH
7D2F 110101             LD       DE,0101H
7D32 D7                 RST      CALBAS
7D33 BA24               DEFW     DRAW#3         ;"DRAW 255,0"
7D35 0100AF             LD       BC,0AF00H
7D38 110101             LD       DE,0101H
7D3B D7                 RST      CALBAS
7D3C BA24               DEFW     DRAW#3         ;"DRAW 0,175"
7D3E 01FF00             LD       BC,00FFH
7D41 11FFFF             LD       DE,0FFFFH
7D44 D7                 RST      CALBAS
7D45 BA24               DEFW     DRAW#3         ;"DRAW -255,0"
7D47 0100AF             LD       BC,0AF00H
7D4A 11FFFF             LD       DE,0FFFFH
7D4D D7                 RST      CALBAS
7D4E BA24               DEFW     24BAH          ;"DRAW 0,-175"
7D50 C3C105             JP       END1           ;ESCE DOPO L'ESECUZIONE
                        END
```

In questa routine ci sono 4 chiamate nella subroutine DRAW 3 che traccia una linea dalla posizione corrente alla posizione distante B-C. Il registro DE contiene rispettivamente SGN y e SGN x.

D R A W n

Il secondo comando spiegato, permette all'utente di sottolineare un certo numero di caratteri. Il comando Basic analogo e':

PRINT OVER 1;"______..._____";

con tanti caratteri "lineetta in basso", quanti sono i caratteri da sottolineare nell'intervallo 1-255.

MODULO DI SINTASSI

Occorre:

- caricare il comando usando (get char); - confrontare con il comando DRAW (FCh/252 dec.);
- procedere attraverso il modulo di sintassi se il codice coincide;
- avanzare CH ADD dopo il comando, usando NEXT CHAR;
- cercare un'espressione numerica usando EXP 1NUM (si aspetta un'espressione che deve rivelarsi numerica);
- chiamare la routine ST END per confermare la fine del comando; cosi' il comando ha superato l'esame di sintassi.

In linguaggio maccchina questo diventa:

```
                    ORG    7D00H          ;32000 DEC.
7D00 D7      EXTEND  RST    CALBAS
7D01 1800            DEFW   GET#CHAR       ;CARICA IL COMANDO
             ;CONFRONTA CON TUTTI I COMANDI IMPLEMENTATI
7D03 FECA            CP     LINE
7D05 CA207D          JP     Z,LINE#SYN     ;PER "LINE"
             ;
7D08 FEFC            CP     DRAW
7D0A CA607D          JP     Z,DRAW#SYN     ;PER "DRAW N"
             ;
7D0D C3F001          JP     ERR#6          ;FINE DELLA RICERCA
             ;MODULO DI SINTASSI DI "DRAW N"
7D60                 ORG    7D60H          ;32096 DEC.
7D60 D7      DRAW#SYN RST   CALBAS
7D61 2000            DEFW   NXT#CHAR       ;AVANZA CH-ADD
7D63 D7              RST    CALBAS
7D64 821C            DEFW   EXPT#1NM       ;CERCA UN'ESPRESSIONE
                                           ;NUMERICA
7D66 CDB705          CALL   ST#END         ;ESCE IN FASE DI CONTROLLO
                                           ;DELLA SINTASSI

7D69 C3C105  DRAW#RUN JP    END1           ;USCITA PROVVISORIA IN ESEC.
                     END
```

Si noti che nella fase di analisi della sintassi la chiamata di EXPT 1NUM esegue soltanto il controllo dell'espressione senza calcolarla. Si noti anche che CH ADD viene avanzato al primo codice dopo l'spressione e il codice e' gia' contenuto nel registro A.

Anche con questo comando il lettore potrebbe accertarsi che il modulo di analisi della sintassi funzioni correttamente prima di inserire il seguente modulo di esecuzione.

MODULO DI ESECUZIONE

```
                    ORG      7D69H             ;32105 DEC.
7D69 D7       DRAW#RUN RST      CALBAS
7D6A 941E              DEFW     FND#INT1       ;N VA NEL REGISTRO A
                                               ;SOLO NELL'INTERVALLO 0-255
7D6C A7                AND      A              ;CONTROLLA SE N=0
7D6D 2005              JR       NZ,DRAW#R1     ;SALTA SE 1<=N<=255
7D6F FD36000A          LD       (IY+ERR#NR),0AH ;MA SE N=0 RITORNA ERRORE
7D73 EF                RST      ROMERR         ;"INTEGER OUT OF RANGE"
              ;GESTISCE OVER 1
7D74 47       DRAW#R1 LD        B,A            ;SALVA N
7D75 3E15              LD       A,OVER         ;CODICE DI CONTROLLO
7D77 D7                RST      CALBAS         ;DI OVER
7D78 1000              DEFW     PRINT#A1       ;"PRINT OVER"
7D7A 3E01              LD       A,01H
7D7C D7                RST      CALBAS
7D7D 1000              DEFW     PRINT#A1       ;DIVENTA "OVER 1"
              ;SOTTOLINEA N CARATTERI
7D7F 3E5F     DRAW#R2 LD        A,UNDLINE
7D81 D7                RST      CALBAS
7D82 1000              DEFW     PRINT#A1       ;SOTTOLINEA UN SOLO
7D84 10F9              DJNZ     DRAW#R2        ;CARATTERE ALLA VOLTA
7D86 C3C105            JP       END1           ;ESCE
```

Si noti che in fase di esecuzione la chiamata della routine EXPT 1NUM calcola l'espressione numerica e sistema il risultato nello STACK del calcolatore, da dove sara' poi raccolto da FIND INT1.

Si noti anche che un messaggio di errore principale viene prodotto caricando la variabbile di sistema ERR NR con il valore appropriato e uscendo con RESTART ROMERR.

C I R C L E   n

Il terzo nuovo comando che viene descritto permette all'utente di disegnare un cerchio di diametro n intorno alla posizione corrente.
In Basic potrebbe essere equivalente a:

```
  10 LET X=PEEK 23677: LET Y=PEE
K 23678
  20 CIRCLE X,Y,n: REM n da spec
ificare
  30 POKE 23677,X: POKE 23678,Y
```

Ma il nuovo comando CIRCLE n, in piu', cattura gli errori di esecuzione e quindi non arresta il programma se, per es., si cerca di disegnare fuori schermo!

MODULO DI SINTASSI

Occorre:

- caricare il codice del comando e confrontarlo con CIRCLE;
- cercare un'espressione numerica;
- chiamare la routine ST END

In linguaggio macchina diventa:

```
     FED8                 CP       CIRCLE
     CA907D               JP       Z,CIRC#SYN  ;PER "CIRCLE N"
      ;MODULO DI SINTASSI DI "CIRCLE N"
                          ORG      7D90H       ;32144 DEC.
7D90 D7          CIRC#SYN RST      CALBAS
7D91 2000                 DEFW     NXT#CHAR    ;AVANZA CH-ADD
7D93 D7                   RST      CALBAS
7D94 821C                 DEFW     EXPT1NUM    ;CERCA UN'ESPRESSIONE NUMERICA
7D96 CDB705               CALL     ST#END      ;ESCE IN F. CONTR. SINTASSI
```

MODULO DI ESECUZIONE

Questo modulo deve comprendere anche una routine di intercettazione degli errori, cioe', in caso di un errore di esecuzione non deve essere visualizzato un messaggio ma deve essere eseguito il comando successivo .
Il listato assembler di questo modulo e':

```
                     ORG      7D99H           ;MOD. DI ESEC. PER "CIRCLE N"
                                              ;32153 DEC.
7D99 2A7D5C   CIRC#RUN LD     HL,(COORDS)
7D9C E5                PUSH   HL              ;COPIA COORDS
7D9D 2A3D5C            LD     HL,(ERR#SP)
7DA0 E5                PUSH   HL              ;COPIA ERR-SP
7DA1 21C77D            LD     HL,ER#ADD
7DA4 E5                PUSH   HL
7DA5 ED733D5C          LD     (ERR#SP),SP     ;INIZIALIZZA LA GESTIONE
7DA9 D7                RST    CALBAS          ;DEGLI ERRORI
7DAA 941E              DEFW   FINDINT1        ;CARICA N
7DAC F5                PUSH   AF              ;SALVA SULLO STACK
7DAD 3A7D5C            LD     A,(COORDS)      ;CARICA LA COORD. X CORRENTE
7DB0 D7                RST    CALBAS          ;SALVA SULLO
7DB1 282D              DEFW   STACK#A         ;STACK DELLA CALCOLATRICE
7DB3 3A7E5C            LD     A,(COORDS+01H)
7DB6 D7                RST    CALBAS
7DB7 282D              DEFW   STACK#A         ;LO STESSO CON COORD. Y
7DB9 F1                POP    AF              ;RICHIAMA N
7DBA D7                RST    CALBAS
7DBB 282D              DEFW   STACK#A         ;ANCHE N SULLO STACK
      ;SULLO STACK DELLA CALCOLATRICE SI TROVANO: X, Y, RAGGIO N,
      ;IL CERCHIO PUO' QUINDI ESSERE DISEGNATO
7DBD D7                RST    CALBAS
7DBE 2D23              DEFW   CIRCLE#1        ;DISEGNA IL CERCHIO
7DC0 E1                POP    HL
7DC1 227D5C            LD     (COORDS),HL     ;RICHIAMA LE COORDINATE
7DC4 C3C105            JP     END1            ;ESCE QUI SE NON CI SONO ERR.
      ;ALTRIMENTI USA LA GESTIONE DEGLI ERRORI
      ;NOTA: LA ROM PRINCIPALE E' IN USO
7DC7                   ORG    7DC7H           ;32199 DEC.
7DC7 E1       ER#ADD   POP    HL
7DC8 223D5C            LD     (ERR#SP),HL     ;RICHIAMA L'INDIRIZZO ORIGINALE
7DCB E1                POP    HL
7DCC 227D5C            LD     (COORDS),HL     ;RICHIAMA LE COORDINATE
7DCF FD3600FF          LD     (IY+ERR#NR),0FFH ;NESSUN ERRORE
7DD3 C9                RET                    ;ESCE SOLO CON RET DATO CHE
                                              ;E' IN USO LA ROM PRINCIPALE
                       END
```

Si notino le fasi della cattura degli errori. Innanzitutto:

- viene memorizzato l'indirizzo corrente di errore (ERR SP);
- viene inserito un nuovo indirizzo di errore nello STACK della macchina;
- ERR SP viene settato per puntare all'indirizzo;

quindi:

- l'indirizzo originale di ERR SP viene richiamato;
- l'errore viene cancellato settando ERR NR a FFh.

B O R D E R * b , i , p

Il quarto nuovo comando permette all'utente di cambiare i colori dello schermo cosi':

```
BORDER colore "b";
INK colore "i";
PAPER colore "p";
```

Questi colori diventano permanenti. In piu' il byte degli attributi dello schermo conterra' il nuovo colore di PAPER. In Basic questo sarebbe:

```
BORDER b:INK i: PAPER p: CLS
```

ma le informazioni sullo schermo non vengono cancellate come con il comando CLS e la posizione di stampa rimane inalterata.

MODULO DI SINTASSI

Occorre:

- caricare il codice del comando e confrontarlo con BORDER;
- confermare la presenza del separatore (*);
- cercare le tre espressioni numeriche separate dalle virgole;
- chiamare la routine ST END.

In linguaggio macchina questo e':

```
     FEE7              CP       BORDER
     CAE07D            JP       Z,BORD*SYN      ;PER "BORDER B,I,P"
           ;MODULO DI SINTASSI PER "BORDER B,I,P"
                       ORG      7DE0H           ;32224 DEC.
7DE0 D7      BORD*SYN  RST      CALBAS
7DE1 2000              DEFW     NXT*CHAR        ;AVANZA CH-ADD
7DE3 FE2A              CP       '*'             ;C'E' IL SEPARATORE?
7DE5 200A              JR       NZ,BORD*ERR     ;SE MANCA SALTA
7DE7 D7                RST      CALBAS
7DE8 2000              DEFW     NXT*CHAR        ;AVANZA CH-ADD
7DEA D7                RST      CALBAS
7DEB 821C              DEFW     EXPT1NUM        ;QUESTO E' "B"
7DED FE2C              CP       COMMA           ;CERCA IL SEPARATORE
7DEF 2802              JR       Z,BORD*S1
7DF1 E7      BORD*ERR  RST       SH*ERR
7DF2 00                DEFB     00H             ;"NONSENSE IN BASIC"
7DF3 D7      BORD*S1   RST      CALBAS
7DF4 2000              DEFW     NXT*CHAR        ;AVANZA CH-ADD
7DF6 D7                RST      CALBAS
7DF7 821C              DEFW     EXPT1NUM        ;QUESTO E' "I"
7DF9 FE2C              CP       COMMA           ;CERCA IL SEPARATORE
7DFB 20F4              JR       NZ,BORD*ERR     ;INDIETRO SE MANCA
7DFD D7                RST      CALBAS
7DFE 2000              DEFW     NXT*CHAR        ;AVANZA CH-ADD
7E00 D7                RST      CALBAS
```

```
7E01 821C            DEFW   EXPT1NUM          ;QUESTO E' "P"
7E03 CDB705          CALL   ST#END            ;ESCE, SINTASSSI OK
                     END
```

Si noti come le tre espressioni numeriche sono caricate in sequenza usando EXPT 1NUM e che i separatori tra le diverse espressioni devono essere virgole. La subroutine BORD ERR usa la routine di restart SH ERR per segnalare un errore che in tempo di esecuzione avrebbe prodotto "Nonsense in Basic".

MODULO DI ESECUZIONE

```
                    ORG     7E06H            ;32262 DEC.
7E06 D7        BORD#RUN RST     CALBAS
7E07 941E               DEFW    FND#INT1         ;P VA IN A
7E09 FE08               CP      08H              ;INTERVALLO CONSENTITO 0-7
7E0B 3005               JR      NC,BORD#R1
7E0D FD360013  BORD#K   LD      (IY+ERR#NR),13H ;"INVALID COLOUR"
7E11 EF                 RST     ROMERR
7E12 07        BORD#R1  RLCA                     ;SPOSTA I BIT
7E13 07                 RLCA
7E14 07                 RLCA
7E15 5F                 LD      E,A              ;SALVA P*B NEL REG. A
7E16 D5                 PUSH    DE               ;E SULLO STACK
          ;ORA CAMBIA TUTTI I BYTE DEGLI ATTRIBUTI
7E17 010003             LD      BC,0300H         ;CI SONO 768 BYTE
7E1A 21FF57             LD      HL,57FFH         ;INDIRIZZO BASE-1
7E1D 23        BORD#R2  INC     HL
7E1E 7E                 LD      A,(HL)           ;CARICA OGNI ATTRIBUTO
7E1F E6C3               AND     0C3H             ;E MODIFICA "PAPER"
7E21 B3                 OR      E                ;BIT DA DARE AL COLORE
7E22 77                 LD      (HL),A           ;SPECIFICATO DA P
7E23 0B                 DEC     BC
7E24 78                 LD      A,B
7E25 B1                 OR      C
7E26 20F5               JR      NZ,BORD#R2       ;ESAURISCE I BYTE
```

```
          ;CONSIDERA IL NUOVO COLORE DI "INK"
7E28 D7            RST    CALBAS
7E29 941E          DEFW   FND#INT1       ;CARICA I
7E2B FE08          CP     08H
7E2D 30DE          JR     NC,BORD#K      ;MESSAGGIO "K" SE > 7
7E2F D1            POP    DE             ;CARICA P*8
7E30 B3            OR     E              ;AGGIUNGE I BIT
7E31 5F            LD     E,A            ;RISULTATO IN E
          ;CAMBIA ATTR-P & ATTR-T
7E32 3A8D5C        LD     A,(ATTR#P)     ;MANTIENE I BIT DI
7E35 E6C0          AND    0C0H           ;"FLASH" E "BRIGHT"
7E37 B3            OR     E              ;AGGIUNGE I & P
7E38 328D5C        LD     (ATTR#P),A     ;RIPRISTINA IL VALORE IN
7E3B 328F5C        LD     (ATTR#T),A     ;ENTRAMBE LE LOCAZIONI
          ;ORA CONSIDERA "B"
7E3E D7            RST    CALBAS         ;LA ROUTINE DEL COMANDO
7E3F 9422          DEFW   BORDER         ;"BORDER" VIENE IMPIGATA
                                         ;VANTAGGIOSAMENTE
7E41 C3C105        JP     END1           ;FINE
                   END
```

Si noti che se viene omesso il separatore X del comando BORDER Xb,i,p, il parametro b (e il comando) puo' essere gestito dall'interprete Basic principale, invece il modulo della sintassi deve comunque considerare tutte le parti del comando anche quelle che vengono ignorate dal modulo di esecuzione.

CONCLUSIONI

I quattro esempi di nuovi comandi dello Spectrum sono soltanto una scusa per familiarizzare con l'argomento. Si augura che il lettore sia in grado di aggiungere i suoi comandi personalizzati. Certo, per ottenere dei buoni risultati, il tempo a disposizione non deve mancare. Per poter lavorare efficientemente in linguaggio macchina, un programmatore attrezzato dovrebbe poter consultare:

"Understanding your Spectrum" di Dr Ian Logan
"Sinclair ZX Spectrum - Assembler e linguaggio macchina per principianti" di W. Tang edito dalla JCE;
"The complete Spectrum ROM disassembly" di Dr. Ian Logan & Dr. Frank O'Hara, edito dalla Melbourne House;
"Alla scoperta dell ZX Spectrum" di Steven Vickers, che e' il manuale di programmazione dello Spectrum,
"Sinclair/Interfaccia ZX1/Microdrive ZX", il manuale di questi dispositivi, edito dalla JCE;

... e avere molto tempo a disposizione.

La Sinclair Research ha aggiunto una nuova spettacolosa dimensione alle capacità del microcomputer Spectrum, il MICRODRIVE.

In questo libro il Dottor Ian Logan, una delle persone più competenti nel campo dei calcolatori Sinclair, fornisce una accurata spiegazione di questo nuovo sistema di memoria di massa ad alta velocità.

Viene spiegato in dettaglio come essa lavora, cosa si può fare in Basic e in Linguaggio Macchina, e quali sono le sue possibilità sia nel campo scolastico che nel campo del lavoro.

Sono inclusi molti programmi, che servono a dimostrare quanto la nuova unità sia versatile.

Nel libro viene inoltre ampiamente trattata l'interfaccia ZX 1, che oltre a consentire il collegamento con i Microdrive, permette di collegarsi in rete con altri Spectrum e anche con altri calcolatori, con altre stampanti e altre periferiche per mezzo dell'interfaccia RS232.

Una caratteristica molto interessante dell'interfaccia ZX 1 è la possibilità che offre ai programmatori esperti di aggiungere nuovi comandi al Basic. Un'ampia discussione su questo argomento, corredata da esempi, conclude il libro.

Chiunque voglia aumentare le possibilità del suo Spectrum, oltre quelle a lui già note, deve leggere questo libro.

COD. 9001

L. 16.000